CONTEXTUALIZED WEB SEARCH: QUERY-DEPENDENT RANKING AND SOCIAL MEDIA SEARCH

A Thesis
Presented to
The Academic Faculty

by

Jiang Bian

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Computing

Georgia Institute of Technology
December 2010

CONTEXTUALIZED WEB SEARCH: QUERY-DEPENDENT RANKING AND SOCIAL MEDIA SEARCH

Approved by:

Dr. Hongyuan Zha, Advisor
College of Computing
Georgia Institute of Technology

Dr. Eugene Agichtein
Mathematics and Computer Science Department
Emory University

Dr. Alexander Gray
College of Computing
Georgia Institute of Technology

Dr. Guy Lebanon
College of Computing
Georgia Institute of Technology

Dr. Richard Vuduc
College of Computing
Georgia Institute of Technology

Date Approved: September 15, 2010

ACKNOWLEDGEMENTS

I would never have been able to finish my dissertation without the guidance of my committee members, help from my friends, and support from my family.

I owe an enormous debt of gratitude to my advisor, Dr. Hongyuan Zha, who has showed me the beauty of science, offered me precious opportunities for collaborations, and supplied sustenance, literally and metaphorically, through long seasons in Atlanta. I would like to express my deep gratitude to one of committee members and collaborators, Dr. Eugene Agichtein, for offering many insightful suggestions and friendly accessibility throughout my graduate study. I would like to thank Dr. Ding Zhou for his patient guidance on my early research work and my PhD life. This dissertation has also received contributions from outside the university, for which I owe my special thanks to Dr. Tie-Yan Liu and Dr. Zhaohui Zheng who mentored my two important industrial internships. I thank Dr. Tao Qin, Dr. Xin Li, Dr. Fan Li and Dr. Max Gubin for giving me so much invaluable advice from industry. I feel especially thankful to my colleagues from Georgia Institute of Technology, Emory University, Microsoft Research Asia, Yahoo! Labs, Facebook, University of Illinois at Urbana-Champaign, and Nokia Research Center, with whom I have enjoyed collaboration.

Finally, I am deeply indebted to my parents for bringing me into this world and providing me with a sound education. Finally, I would like to thank Dr. Weiyun Chen, who was always standing by me through both good and bad times.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Due to the information explosion on the Internet, effective information search techniques are required to retrieve the desired information from the Web. With much analysis on users' search intention and the variant forms of Web content, we find that both the query and the indexed web content are often associated with various context information, which can provide much essential information to indicate the ranking relevance in Web search. Although there have been many existing studies on extracting the context information of both the query and the Web content, little research has addressed exploring these context information to improve Web search. This dissertation seeks to develop new search algorithms and techniques by taking advantage of rich context information to improve search quality.

This dissertation consists of two major parts. In the first one, we study how to explore the context information of the query to improve search performance. Since Web queries are usually very short, it is difficult to extract precise information need from the query itself. We propose to take advantage of the context information, such as the search intention of the query, to improve the ranking relevance. According to the query difference in terms of search intention, we first introduce the query-dependent loss function, by optimizing which we can obtain better ranking model. However, in practical search engine, it is uneasy to precisely define the query-dependent loss function. And, inspired by the requirement of deep dive and incremental update on dedicated ranking models, we investigate a divide-and-conquer framework for ranking specialization. Experimental results on a large scale data set from a commercial search engine demonstrate significant improvement on search performance over currently applied ranking models without considering query context.

The second part of this dissertation investigates how to extract the context of specific Web content and explore them to build more effective search system. This study focuses on searching over social media, the new emerging form of Web content. As the fastest growing segment of the Web, social media services establish new forums for content creation Daily, huge amount of social media content are collaboratively generated by millions of Web users, driven by various of social activities. Due to the valuable information contained in the resulting archives of both the content and the context of the interactions, computational methods for knowledge acquisition has become an important topic in social media analysis. Unlike traditional Web content, social media content is inherently associated with much new types of context information, including content quality, user reputation, and user interactions, all of which provide useful information for acquiring knowledge from social media. In this dissertation, we seek to develop algorithms and techniques for effective knowledge acquisition from collaborative social media environments by using the dynamic context information. In particular, this study first proposes a new general framework for searching social media content, which integrates both the content features and the user interactions. Then, a semi-supervised framework is proposed to explicitly compute content quality and user reputation in social media. These new context information are incorporated into the general search framework to improve the search quality. Experimental results of large scale evaluation on real world social media content demonstrate that this research achieves significant improvements over previous approaches for information search in social media. Furthermore, this dissertation also investigates techniques for extracting the structured semantics of social media content. Experimental results demonstrate that this kind of context information is essential for improving the performance of content organization and retrieval over social media service.

CHAPTER I

INTRODUCTION

The task of Web search can be briefly described as follows. When Web user intends to find some information from a collection of indexed Web content, she first generate a query to represent her information need. Then, given the query, the deployed ranking model measures the relevance of each indexed Web content to the query, and sorts them based on the relevance scores, and finally presents a list of top-ranked Web content to the user. Based on this big picture of Web search, we find that ranking model plays an essential role in an effective search system. Thus, this study focuses on building the ranking model which can precisely indicate the relevance between the query and the indexed Web content.

However, there are many challenges for obtaining the effective ranking model. First, since queries are usually very short, whose average length is less than three according to the recent study on several popular commercial search engines, it is uneasy to understand the user's information need from two or three words. In addition, it is challenging to analyze the Web content. As many new emerging types of Web content, such as social media content, become more popular on the Web, it is not able to use the single method to analyze various types of Web content. Furthermore, we can not analyze the content quality only based on the text of Web content.

With much analysis on users' search intention and the variant forms of Web content, we find that both the query and the indexed web content are often associated with much context information. For example, beyond the text itself, queries can be associated with the context of search intention, which indicates the query is navigational, informational or transactional; queries can also be associated with the context

of semantics, which specifies whether the user looks for products, travel advice or any other information; moreover, queries have other context information, including length, popularity, etc. These various types of context information can be used to better understand the users' information need so as to improve the search performance.

Furthermore, the indexed Web content is also associated with context information such as hyperlinks between Web pages which can be used to compute the importance of each Web page. Additionally, the author's reputation and the evaluation or ratings from other Web users are other valuable context information for indicating the quality of Web content. Based on these observation, we find that the context of queries and Web content can provide much essential information to indicate the ranking relevance in Web search.

Despite many existing studies on extracting the context information of both the query and the Web content, little research has addressed exploring these context information to improve Web search. Motivated by these consideration, this dissertation seeks to develop new search algorithms and techniques by leveraging rich context information to improve search quality. In particular, this dissertation consists of two major parts. The first one studies the context of the query in terms of various ranking objectives of different queries, and proposes to incorporate such query context information into the ranking model to improve the ranking relevance. The second part investigates how to extract the context of specific Web content and explore them to build more effective search system. This study is based on the new emerging form of Web content, social media content, which is inherently associated with much new context information, including content quality, user reputation, and user interactions. In this dissertation, we seek to develop algorithms and techniques required for effective knowledge acquisition from collaborative social media environments by using the dynamic context information. In the following, we will give a brief introduction of

each part, respectively.

1.1 Query-Dependent Ranking for Web Search

Queries describe the users' information need and therefore they play an essential role in the context of ranking for information retrieval and Web search. However, most of existing approaches for ranking do not explicitly take into consideration the fact that queries vary significantly along several dimensions and entail different treatments regarding the ranking models. To this end, this study considers the query difference in terms of search intention as new context information, and proposes to incorporate such context information into ranking. In this dissertation, we explore two general approaches, query-dependent loss approach and rank specialization approach.

In the context of Web search, according to search intention, queries are usually classified according to search intent such as navigational, informational and transactional queries. Based on the observation that such kind of query categorization has high correlation with the user's different expectation on the result accuracy on different rank positions, this study first explores the approach of developing position-sensitive query-dependent loss functions exploring such kind of query categorization. Beyond the simple learning method that builds ranking functions with pre-defined query categorization, this study further proposes a new method that learns both ranking functions and query categorization *simultaneously*. This query-dependent loss approach is applied to several existing ranking algorithms. And, experimental results demonstrate that query-dependent loss functions can be exploited to significantly improve the accuracy of learned ranking functions. It is also shown that the ranking function jointly learned with query categorization can achieve better performance than that learned with pre-defined query categorization. Finally, this study also provides analysis and conduct additional experiments to gain deeper understanding on the advantages of ranking with query-dependent loss functions over

other query-dependent and query-independent approaches.

For the ranking approach with query-dependent loss function, however, we only consider very few number of query categories. In practical search engine, queries can be grouped into more finer topics, and it is very difficult to precisely describe the query difference into the loss function. Moreover, the commercial search engines usually require deep dive and incremental update on dedicated ranking model. Motivated by these considerations, this dissertation proposes a divide-and-conquer framework for ranking specialization, i.e. learning multiple ranking models by addressing query difference. Specifically, this approach first generates query representation by aggregating ranking features through pseudo feedbacks, and employ unsupervised clustering methods to identify a set of ranking-sensitive query topics based on training queries. To learn multiple ranking models for respective ranking-sensitive query topics, a global loss function is defined by combining the ranking risks of all query topics, and a unified learning process is introduced to minimize the global loss. Moreover, an ensemble approach is employed to generate the ranking result for each test query by applying a set of ranking models of the most appropriate query topics. Experiments are conducted on a benchmark dataset for learning ranking functions as well as a dataset from a commercial search engine. Experimental results show that this approach can significantly improve the ranking performance over existing single-model approaches as well as straightforward local ranking approaches, and the automatically identified ranking-sensitive topics are more useful for enhancing ranking performance than pre-defined query categorization.

1.2 Context-Aware Social Media Search

The second part of this dissertation investigates how to extract the context of specific Web content and explore them to build more effective search system. This work will majorly focus on searching over the new emerging online social media service.

Online social media services comprise one of the fastest growing segments on the Web. They establish new forums for content creation, allow Web users to connect to each other and share information, and permit novel social applications at the intersection of people and information. Such content includes community question answering (Yahoo! Answers [1]), social bookmarking (Delicious [2]), social encyclopedia (Wikipedia [3]) social photo sharing (Flickr [4]), social video sharing (Youtube [5]), microblogging (Twitter [6]), and many other form of user-generated content. During the last few years, this new kind of content has started dominating the Web: increasingly, Web users participate in content creation, rather than just consumption. Published and professional Web content together is estimated to be generated at about 5 Gigabyte per day, whereas user-generated content is created at the rate of about 10 Gigabytes a day, and growing [63]. As social media services grow in size and popularity, the resulting archives of both the content and the context of the social interactions contain valuable information, which can enable new knowledge-rich approaches to information access.

Unlike traditional Web content, social media content is inherently associated with much new context information, including content quality, user reputation, and user interactions, all of which provide useful information for acquiring knowledge from social media. In this dissertation, we seek to develop algorithms and techniques required for effective knowledge acquisition from collaborative social media environments by using the dynamic context information.

Despite many existing information retrieval approaches applied to traditional Web content, finding information and extracting knowledge from social media content is

[1] http://answers.yahoo.com/
[2] http://del.icio.us/
[3] http://wikipedia.com/
[4] http://flickr.com/
[5] http://youtube.com/
[6] http://twitter.com/

a challenge task. First, unlike traditional Web content, social media content is inherently dynamic: the perceived popularity, the availability of interactions, the organization of content, or even interpretation of the content may change significantly over time. Furthermore, the quality of social media content varies even more than the quality of traditional Web content. For example, a large fraction of social media content often reflects unsubstantiated opinions of users, which are not useful for mining knowledge. However, social media content is inherently associated with much new context information, including content quality, user reputation, and user interactions, all of which provide useful information for acquiring knowledge from social media. Motivated by these considerations, this dissertation seeks to develop algorithms and techniques required for effective knowledge acquisition from collaborative social media environments by using the dynamic context information.

To make collaboratively generated content in social media more accessible and serving a broad spectrum of users, it is important to provide an effective search interface. As discussed, social media content is different from traditional Web content in style, quality, and authorship. More importantly, the explicit support for social interactions between users, such as posting comments, rating content, and responding to questions and comments makes social media unique and requires new techniques for search. This work uses searching community questions answering (CQA) archives as a concrete example to discuss the various issues involved in searching collaboratively generated content.

Community question answering (CQA) has emerged as a popular and effective paradigm for a wide range of information needs. For example, to find out an obscure piece of trivia, it is now possible and even very effective to post a question on a popular CQA site such as Yahoo! Answers, and to rely on other users to provide answers, often within minutes. The importance of such CQA sites is magnified as they create archives of millions of questions and hundreds of millions of answers, many of

which are invaluable for the information needs of other searchers. However, to make this immense body of knowledge accessible, effective answer retrieval is required. In particular, as any user can contribute an answer to a question, the majority of the content reflects personal, often unsubstantiated opinions. A ranking that combines both relevance and quality is required to make such archives usable for factual information retrieval. This task is challenging, as the structure and the contents of CQA archives differ significantly from the web setting. To address this problem, this study proposes a general ranking framework for factual information retrieval from social media. Further result analysis is also provided to gain deeper understanding of which features are significant for social media search and retrieval.

In addition, as the submitted questions and answers are collaboratively generated by users, the quality of such content varies widely - increasingly so that a large fraction of the content is not usable for answering queries. Previous approaches for retrieving relevant and high quality content have been proposed, but they require large amounts of manually labeled data – which limits the applicability of the supervised approaches to new sites and domains. This study address this problem by developing a semi-supervised coupled mutual reinforcement framework for simultaneously calculating content quality and user reputation, that requires relatively few labeled examples to initialize the training process. More importantly, this quality estimation can be incorporated into the proposed ranking framework to improve the accuracy of search over CQA archives.

Another issue involved in search CQA archives is related to social interactions. As online social media draws heavily on active reader participation, such as voting or rating of news stories, articles, or responses to a question, this user feedback is invaluable for ranking, filtering, and retrieving high quality social media content. Unfortunately, as social media moves into the mainstream and gains in popularity, the quality of the

user feedback degrades. Some of this is due to noise, but, increasingly, a small fraction of malicious users are trying to "game the system" by selectively promoting or demoting content for profit, or fun. Hence, the proposed ranking framework must be robust to noise in the user interactions, and in particular to vote spam. Therefore, this study considers several vote spam attacks, and introduce a method of training the proposed ranker to increase its robustness to some common forms of vote spam attacks.

Results of a large scale evaluation demonstrate that the proposed ranking framework in this dissertation is highly effective at retrieving well-formed, factual answers to questions, as evaluated on a standard factoid question answering benchmark, and that the proposed quality estimation method are more effective than previous approaches for finding high-quality answers, questions, and users such that it improves the accuracy of search over CQA archives over the state-of-the-art methods. Moreover, the experimental results show that the proposed ranking framework is significantly more robust to vote spam compared to a state-of-the-art baseline as well as the ranker not explicitly trained to handle malicious interactions.

Furthermore, this work investigates techniques for extracting the structured semantics of social media content, especially on task questions in CQA. Recently, task questions, as a specific type of information need, have been widely used by Web users in the context of Web search. Fortunately, CQA provides an alternative channel for solving task questions, since the large number of task questions posted in CQA sites have comprised a valuable knowledge repository which could be a gold mine for automatic task solving. This work proposes to use a generative topic modeling approach for comparative text mining to extract structured semantics of task questions in a principled way. We further propose to incorporate prior expert knowledge into the learning process. Empirical experiments are carried out on integrating questions into task knowledge for two different general task topics from a CQA portal. The results

show that the proposed method is effective for both extracting structured semantics of questions and generating integrated task summaries for each task topic. Additional experimental results also demonstrate that this kind of context information is essential for improving the performance of content organization and retrieval over social media service.

1.3 Contributions

In summary, this dissertation explores using the context information of both the query and the Web content to improve the search performance. It first investigates the query context in terms of different search intention, and studies how to incorporate such query context information into the ranking model. The contributions on this objective can be summarized as:

- Proposing to incorporate query difference into ranking by introducing query-dependent loss functions, and developing a new methods for learning the ranking function jointly with learning query difference in terms of query categorization (Chapter 3).

- Applying a divide-and-conquer framework for ranking specialization, which learns multiple ranking models by addressing the difference between multi-type content with multi-type queries (Chapter 4 in progress).

Secondly, this dissertation studies how to extract the context of the specific Web content and use the obtained context information to improve search performance. In particular, this study focuses on searching over the social media contents, especially the Community Question Answering. The contributions on this direction can be concluded as:

- Introducing an effective framework or learning ranking functions for question answering that incorporates community based features and user preference (Chapter 5).

- Proposing a semi-supervised mutual reinforcement framework to calculate the quality and reputation scores of multiple sets of entities in network relationship, simultaneously (Chapter 6).

- Designing a parameterized vote spam model to describe and analyze common forms of vote spam in social media and proposing a method for increasing the robustness of ranking over social media content by injecting noise at training (Chapter 7).

- Defining a new problem of extracting structured semantics of task questions and generating integrated task summaries, and employing a probabilistic cross-collection mixture modeling approach for addressing the problem (Chapter 8).

1.4 Dissertation Organization

The remaining parts of this study are organized as follows: First, Chapter 2 introduces the related work. As introduced in Chapter 1, this dissertation consists of two major parts. (1) In Part 1 of query-dependent ranking for Web search: Chapter 3 initiates focus on incorporating query difference into the loss function and proposes a new method for learning ranking functions jointly with query difference. Chapter 4 extends the research by applying a divide-and-conquer framework for ranking specialization. (2) In Part 2 of context-aware social media search: Chapter 5 presents new general framework for information retrieval over social media, especially on community question answering. Chapter 6 proposes a semi-supervised coupled mutual reinforcement framework for simultaneously recognizing reliable users and content

in social media. Chapter 7 designs a parameterized vote spam model in social media, and introduces a new training method for increasing the robustness of ranking over social media against common forms of vote spam. Chapter 8 addresses the new problem of extracting structured semantics for task questions in CQA and generating integrated task summaries. Finally, conclusions are drawn in Chapter 9.

CHAPTER II

RELATED WORK

This dissertation has three main categories of related work, the first category corresponding to previous research on learning to rank and query-dependent ranking approaches while the second and the third category corresponding to previous studies on community question answering services, especially on the issues of information retrieval and identification of reliable content and users over the social media environment as well as information extraction and integration in the context of social media services.

2.1 Ranking for Web Search

Ranking has become an essential research issue for informational retrieval and Web search, since the quality of a search system is mainly evaluated by the relevance of its ranking results. The task of ranking in the search process can be briefly described as follows. Given a query, the deployed ranking model measures the relevance of each document to the query, sorts all documents based on their relevance scores, and presents a list of top-ranked ones to the user. Thus, the key problem of search technology is to develop a ranking model that can best represent relevance.

2.1.1 Learning to Rank

Many models have been proposed for ranking, including the Boolean model [4], vector space model [67], probabilistic model [64] and language model [42, 60]. Recently, there are renewed interests in exploring machine learning methodologies for building ranking models. Many learning-based approaches have been introduced, some popular examples of which include MCRank [45], RankNet [16], RankSVM [35], RankBoost [22],

GBRank [82], ListNet [19], ListMLE [77], and IsoRank [83]. These approaches leverage training data, which consists of queries with their associated documents and relevance labels, and machine learning techniques to make the tuning of ranking models theoretically sound and practically effective.

2.1.2 Incorporate Query Difference in Ranking

Some of recent works have realized the importance and necessity of incorporating query difference into learning the ranking function. Zha et al. [79] propose an aTVT algorithm which implicitly incorporates query difference using monotonic transformations of the learned ranking functions. This approach focuses on the boundary of each query without considering broader query grouping. Kang et al. [38] classify queries into two categories (navigational and informational) and build two corresponding ranking models separately. However, it requires the availability and high accuracy of query classification. In a most recent work, Geng et al. [24] propose a K-Nearest Neighbor (KNN) method to employ different ranking models for different queries. Specifically, each training query holds a ranking model which is learned using the query itself and its neighboring queries. Given a test query, they find the most similar training query and use the corresponding model for ranking. Training time of this method is quite large, since many models need to be trained separately. And each model is trained using only a part of whole training set, which may cause the declining accuracy due to the lack of adequate training examples. In order to avoid such problems, this study explores two approaches for incorporating query difference into ranking. The first one is to introduce query-dependent loss functions and learn the ranking functions jointly with query categorization [7]. Due to some restriction in practical search engine, we propose another approach that employs a divide-and-conquer approach for ranking specialization to improve the ranking relevance [6].

2.2 Search in Social Media

Recently, many emerging types of services and their associated content have become more popular on the Web, one of which is the online social media service that provides many popular web applications such as photo sharing (Flickr), social bookmarking (Delicious), and video sharing (Youtube), and, more recenlty, popular Community Question Answering (CQA). Question answering over CQA archives is different from traditional TREC QA [58], and applying QA techniques over the web [14]. The most significant difference is that traditional QA operates over a large collection of documents (and/or web pages) whereas this study is attempting to retrieve answers from a social media archive with a large amount of associated user-generated metadata [3]. This metadata (such as explicit user feedback on answer quality) is crucial due to the large disparity of the answer quality, as any user is free to contribute his or her answer for any question.

2.2.1 Information Retrieval over Community Question Answering

Due to the explosive rise in popularity of Yahoo! Answers and Naver and other sites, CQA has recently become an active area of research. This area of QA can be traced to the research on answering questions using Usenet Frequently Asked Questions (FAQ) archives [17, 71, 44, 72]. Usenet FAQs could be viewed as precursors to today's CQA archives that fulfilled a similar role but lacked intuitive support for explicit user feedback and other user interactions. More recently, Jeon et al. [33] presented a machine translation model to find similar questions from a CQA service, but did not take quality of answers into consideration. Su et al. [74] analyzed the quality of answers in CQA portals. Jeon et al. [34] built a model for answer quality based on features derived from the specific answer being analyzed. Agichtein et al. [3] presented a supervised approach to mining user interaction and content-based lexical features to identify high-quality content in CQA. Recently, Bian et al. [9] developed

a ranking system to retrieve relevant and high-quality answers. While these models have shown to be quite effective for finding high quality [3] and relevant [9] content, they do not explicitly model user reputation, and require substantial amounts of manual supervision.

2.2.2 Modeling User Authority

At the same time, there has been a long-standing interest in modeling authority, reputation and expertise in social networks and communities. Link-based ranking algorithms have been shown to be successful in the context of evaluating quality of Web pages. Two of the most prominent link-analysis algorithms are PageRank [59] and HITS [40]. Variations of PageRank and HITS have already been applied in many contexts, especially for propagating reputation and finding experts in the mutual reinforcement process. Guha et al. [25] and Ziegler [87] study the problem of propagating trust and distrust among users in social media, while considering trust as a transitive property in network relationships. Expert finding is also an active area of research, where researchers also take advantage of mutual reinforcement principle. Zhang et al. [81] analyze data from an on-line forum, seeking to identify users with high expertise. They apply both ExpertiseRank and HITS to identify users with high expertise. Jurczyk and Agichtein [37] show an application of the HITS algorithm to a CQA portal, especially the user interactions graph, and show a positive correlation between authority calculated with the HITS algorithm and answer quality. Campbell et al. [18] compute the score of HITS over the user-user graph in a network of e-mail exchanges, showing that it is more correlated to quality than other metrics. Zhou et al. [86] propose a method for co-ranking authors and their publications using their networks. Dom et al. [21] also study the performance of several link-based algorithms to rank people by expertise on a network of e-mail exchanges.

Although link-based and probabilistic approaches have been shown to be successful in ranking entities on the graph, most of them focus on ranking only one type of entity, and few of them utilize other properties of the entities except with link structure. Building on the previous work, this study proposes a new framework which is based on the model of network relationships in social media [69] and exploits the mutual reinforcement principle [78]. Particularly, this study proposes a mutual reinforcement framework for ranking sets of entities [10], specifically applied to the CQA network that connects users, questions, and answers. This approach take advantage of mutually reinforcing relationship to rank various sets of entities simultaneously, and in this approach, many other features are used besides link structure.

2.2.3 User Interactions and Related Spam in Web Search

This work is partly similar in spirit to integrating user interactions and feedback into web search [35, 36, 39, 2]. For example, implicit feedback in the form of result clickthrough was shown to be helpful for web search ranking. The main difference of this work is that it focuses on question answering, which is a more precise form of search compared to general-purpose web search explored in the past. As another departure from previous work, this work does not assume the existence of large amounts of expertly labeled relevance judgments, but instead automatically generate relevance labels. These differences in setting provide an interesting challenge for learning ranking functions over CQA.

However, one of the serious problems when integrating user interactions to web search is click spam [30]. Many studies have analyzed robustness of web search ranking to click spam. Radlinski et al. [62] presented how click noise/spam bias the ranking results. Jansen [30] revealed the influence of malicious clicks on online advertising search and Metwally et al. [57] explored how to identify fraudulent clicks on advertisements. After a deep analysis on click fraud in online advertising, Immorlica

et al. [29] demonstrated that a particular class of learning algorithms are resistant to click fraud in some sense. Radlinski et al. [61] analyzed click spam from a utility standpoint and investigated whether personalizing web search results can reduce spam. Several previous studies explored interactions spam in social systems. Heymann et al. [27] surveyed the approaches and challenges for fighting spam on social web sites. Mehta et al. [53] provided an algorithm for detecting spam in collaborative filtering. This study focuses on a different setting of ranking in social media, and considers general methods of vote spam which exhibit distinct attack methods and characteristics from click fraud [8]. This study also explicitly validates the robustness of the proposed ranking method in a community question answering setting.

2.2.4 Information Extraction and Integration over Social Media

Beyond the content quality and user reputation, there are still much useful context information which can be extracted from social media based on information extraction and integration techniques. This section reviews several lines of work which are closely related to the approaches for information extraction and integration in social media.

Related work on document content characterization [12, 66, 28, 52, 73, 85] introduce a set of probabilistic models to simulate the generation of a document. Several factors in producing a document, either observable (e.g. author [66, 73]) or latent (e.g. topic [12, 28, 52]), are modeled as variables in the generative Bayesian network and have been shown to work well for document content characterization.

Two popular topic analysis approaches, the Probabilistic Latent Semantic Analysis (PLSA) [28] and the Latent Dirichlet Allocation (LDA) model [12], are based upon the idea that the probability distribution over words is a document can be expressed as a mixture of topics, where each topic is a probability distribution over words. Along the line of LDA, the Author-Word model proposed in [52] considers the interests of single authors as the origin of a word. Influential following work named

Author-Topic model combines the Topic-Word and Author-Word models, such that it regards the generation of a document as affected by both factors in a hierarchical manner [66, 73]. A recent work on social network analysis [85] extends the previous model with an additional layer that captures the community influence in the setting of information society. The model proposed in this work [84] is different from the Author-Topic model, where the users or sources of the tags and documents are observed instead of being latent.

The line of PLSA has been widely and successfully applied to blog articles and other user-generated text collections to mine topic patterns [46]. In particular, Mei et al. [55] employed the probabilistic topic model to extract the subtopics in weblog collections, and track their distribution over time and locations; lately, they also proposed a mixture topic model to model both facets and opinions at the same time [54]. Recently, Lu et al. [49] presented a semi-supervised topic model to solve the problem of integrating opinions expressed in a well-written expert review with those scattering in blog spaces and forums. A more recent work [50] studied the problem of generating rated aspect summary by using the topic model to model the dependency structure of phrases in user-generated short comments. All of these studies applies the topic model for clustering on the single text collection.

One of directions in this work focuses on discovering latent aspects across all task instances under the same task topic [11], which is like a comparative text mining problem. Some previous works have address this problem by using couple clustering method [51] or using cross-training for learning classifier from multiple document sets [68]. This study models the structured semantics of CQA questions using a cross-collection mixture model [80], and extends this model by incorporating prior knowledge on question semantics from experts.

CHAPTER III

QUERY-DEPENDENT RANKING FOR SEARCH

3.1 Query Difference in Ranking

As discussed in the above chapters, ranking has become an essential research issue for Web search, and query plays an important role in the task of ranking. But, queries vary largely in the context of Web search. Previous studies have introduced many query taxonomies, such as that in terms of semantics [5, 70] and that in terms of search intentions [43, 65]. In particular, for query taxonomy of semantics, queries can be classified into product queries, travel queries, and other semantic groups; while for query taxonomy of search intention, queries can be coarsely categorized as navigational, informational and transactional [15].

In most of previous works, the significant difference in queries are not adequately addressed in the context of ranking, which is clearly not appropriate. In this chapter, we propose a new query-dependent ranking approach that improve the search performance by introducing the query different into the ranking algorithms.

Although query difference is multi-faceted, we observe that query difference usually has tight correlation with the user's different expectation on the result accuracy on different rank positions. Let us elaborate this issue using Broder's "Taxonomy of Web search" [15], which describes query difference based on the search intent of users and classifies queries into three categories: navigational, informational and transactional. In particular, navigational queries are those which are intended to find a specific Web site that the user has in mind; informational searches are intended to find information about a topic; transactional ones are intended to complete some Web-mediated activities. Therefore, for the navigational and transactional query, the

user expects high accuracy on the top one retrieved result; while for the informational query the user looks for more relevant documents among top-K rank positions.

This kind of position-sensitive query difference requires respective objectives for the ranking model. Specifically, for the navigational and transactional query, the ranking model should aim to rank the exact Web page that the user is looking for on the top position of the result list; while for the informational query, the ranking model should target at presenting a set of Web pages relevant to the topic of the query on the top-K positions of returned results. The above only illustrates one aspect of the issue, we can similarly consider the issue in the context of subtopic retrieval and topic distillation. In particular, for the subtopic retrieval query, the objective of ranking model should be presenting a set of Web pages covering as many subtopics as possible on the top-K positions of result list; while for topic distillation query, the ranking model should focus on ranking a set of Web pages best representing one single topic among top-K rank positions. Motivated by these observation, it is essential to take into account of query difference for defining different ranking objectives in the process of learning to rank.

In this chapter, we propose to incorporate query difference into ranking by introducing query-dependent loss functions in the learning process. Inspired by the diverse ranking objectives implied by various queries, we apply different loss functions to different queries in learning the ranking function. Since it is difficult and expensive in practice to extract individual objective for each query, we make use of query categorization to represent query difference such that each query category stands for one kind of ranking objective. In this study, we focus on Broder's taxonomy of Web search [15] and develop position-sensitive query dependent loss functions according to this popular query categorization

Unfortunately, query categorization may or may not be available at learning time. Accordingly, beyond learning the ranking functions with pre-defined query categories,

we develop a new method for learning ranking functions jointly with query categorization without prior knowledge on query categorization. In this new method, the ranking function and query categorization use totally disjointed feature sets.

To evaluate the effectiveness of our proposed approach, we derive the position-sensitive query-dependent loss function based on Broder's taxonomy, and apply it to two popular ranking algorithms, RankNet and ListMLE. Experimental analysis is employed to verify that query-dependent loss function can be exploited to boost the accuracy of ranking for Web search. We also make a comparison on the ranking accuracy between the learning method that trains the ranking function with predefined query categorization and that learns the ranking function jointly with query categorization. Moreover, we provide analysis and conduct additional experiments to gain deeper understanding on the advantages of ranking with query-dependent loss functions over other query-dependent or query-independent ranking approaches.

3.2 Incorporating Query Difference into Ranking

In this section, we propose to incorporate query difference into ranking by introducing query-dependent loss function, and we also outline learning methods for ranking with the query-dependent loss functions.

3.2.1 Query-Dependent Loss Functions

We formalize the problem of building a ranking model as finding a function $f \in \mathcal{F}$, where $\mathcal{F}$ is a given function class, such that f minimizes the risk of ranking in the form of a given loss function L_f. For a general learning to rank approach, the loss function is defined as:

$$L_f = \sum_{q \in Q} L(f), \tag{1}$$

where Q denotes the set of queries in the training data; $L(f)$ denotes a query-level loss function, which is defined on ranking function f and has the same form among

all queries.

Inspired by the diverse ranking objectives implied by the queries, we incorporate query difference into the loss function by applying different loss functions to different queries. This kind of query-dependent loss function is defined as:

$$L_f = \sum_{q \in Q} L(f; q), \tag{2}$$

where $L(f; q)$ is the query-level loss function defined on both query q and ranking function f, and each query has its own form of loss function.

However, it is difficult and expensive in practice to define individual objective for each query. Thus, we take advantage of query categorization to represent query difference, which means each query category stands for one kind of ranking objective. In general, we assume there is a query category space, denoted as $\mathbf{C} = \{\mathcal{C}_1, \cdots, \mathcal{C}_m\}$, where $\mathcal{C}_i (i = 1, \cdots, m)$ represents one query category. We also assume a soft query categorization, which means each query can be described as a distribution over this space. We use $P(\mathcal{C}_i|q)$ to denote the probability that query q belongs to the class $\mathcal{C}_i$ with $\sum_{i=1}^{m} P(\mathcal{C}_i|q) = 1$. Thus, the query-dependent loss function of the ranking function f is defined as:

$$\begin{aligned} L_f &= \sum_{q \in Q} L(f; q) && (3) \\ &= \sum_{q \in Q} \left(\sum_{i=1}^{m} P(\mathcal{C}_i|q) L(f; q, \mathcal{C}_i) \right), && (4) \end{aligned}$$

where $L(f; q, \mathcal{C})$ denotes a category-level loss function defined on query q, ranking function f and q's category $\mathcal{C}$.

3.2.2 Learning Methods

After constructing the query-dependent loss function by incorporating the information of query categories, we can learn the ranking function by minimize the loss function. The straightforward method is to first obtain the soft categorization for

each query, i.e. $P(\mathcal{C}_i|q), i = 1, \cdots, m$, then to learn the ranking function by minimizing the query-dependent loss function (Eq. 4) with known query categorization. In this simple method, the query categorization is obtained independently with learning the ranking function. However, query categorization may not be precise enough and even not be available at the learning time. Thus, we propose a new learning method which considers query categorization as hidden information and optimizes the ranking function jointly with query categorization. Compared to the straightforward method, this method aims to categorize queries for the purpose of minimizing the loss function for ranking. We denote this method as the *unified* method. We will discuss how to specify the learning method with respect to particular query-dependent functions in the following section.

3.3 Applying Query-Dependent Loss To A Specific Query Categorization

In terms of search intentions, queries are usually classified into three major categories according to Broder's taxonomy [15]: navigational, informational, and transactional. Under this taxonomy, a navigational query is intended to locate a specific Web page, which is often the official homepage or subpage of a site; an informational query seeks information on the query topic; and a transactional query seeks to complete a transaction on the Web.

For this taxonomy of Web search, we observe that the rank objectives usually have tight correlation with the user's different expectation on the result accuracy on different rank positions. Specifically, for the navigational and transactional query, the ranking model should aim to retrieve the exact relevant Web page on the top one position in the result sets; while for the informational query, the ranking model should target to rank more relevant Web pages on a set of top positions in the result sets. To incorporate this kind of query difference into ranking, we can define the position-sensitive query-dependent loss function by targeting the ranking objectives

to respective sets of ranking positions for different query categories. In particular, for the navigational and transactional query, the loss function should focus on that exact relevant page; while for the informational query, the loss should consider those relevant pages which should be ranked in the range of top-K positions.

In this section, we first define the query-dependent loss functions which represent these position-sensitive objectives. Then, we introduce the learning methods to minimize this position-sensitive query-dependent loss function. Moreover, we present two examples of applying the proposed position-sensitive query-dependent loss function to two concrete ranking algorithms, respectively.

3.3.1 Query-Dependent Loss Functions Based on Query Taxonomy of Web Search

According to Broder's taxonomy and corresponding ranking objectives discussed above, we classify queries into two categories, i.e., $\mathbf{C} = \{\mathcal{C}_I, \mathcal{C}_N\}$, where $\mathcal{C}_I$ denotes informational queries and $\mathcal{C}_N$ denotes navigational and transactional queries. Note that we combine navigational and transactional queries into $\mathcal{C}_N$ since both of them describe the similar search intention which focuses on the accuracy of top-one ranked result.

According to Eq. 4, the query-dependent loss function is now defined as:

$$L(f; q) = \alpha(q)L(f; q, \mathcal{C}_I) + \beta(q)L(f; q, \mathcal{C}_N), \tag{5}$$

where $\alpha(q) = P(\mathcal{C}_I|q)$ represents the probability that q is an informational query, $\beta(q) = P(\mathcal{C}_N|q)$ indicates the probability that q is a navigational or transactional query, and $\alpha(q) + \beta(q) = 1$.

In general, we can define a position-sensitive loss function $L(f; q, \mathcal{C})$ in the form of:

$$L(f; q, \mathcal{C}) = \sum_{x \in X_q} l(f(x), g(x), p(x); \Phi(q, \mathcal{C})) \tag{6}$$

where X_q is the set of training examples under query q; and x is one of the training examples; the example-level loss l is defined based on ranking function f, ground truth of the example $g(x)$, the true ranking positions of the example $p(x)$ [1], and a set of ranking positions $\Phi(q, \mathcal{C})$ on which users expect high result accuracy. The general principle is that the example-level loss l will contribute to the whole loss if the true rank position $p(x)$ of the example x is included in $\Phi(q, \mathcal{C})$, and the actual value of example-level loss is defined by $f(x)$ and $g(x)$. Specifically, we assume that, the ranking risk of informational query, $L(f; q, \mathcal{C}_I)$, focuses on documents that should be ranked on top-k_I positions, i.e., $\Phi(q, \mathcal{C}_I) = \{1, \cdots, k_I\}$; while the ranking risk of navigational or transactional query targets on documents that should be ranked on top-k_N positions, i.e., $\Phi(q, \mathcal{C}_N) = \{1, \cdots, k_N\}$.

3.3.1.1 Estimating Rank Positions from Relevance Judgments

As discussed above, in order to build the query-dependent loss function, we need to obtain the true rank position of each training example. The relevance judgments in the training set provide the possibility to obtain the true rank position of each training example. Multi-level relevance judgments are often used in the training set. For example, if all the training examples are labeled using a k-level relevance judgment, the label set contains k distinct relevance levels, such as $\{0, 1, \cdots, k-1\}$, where larger value usually indicates higher relevance.

However, there is an apparent gap between the true rank positions and multi-level relevance judgments. In particular, for some queries, more than one documents may have the same label. In such case, any document x, sharing the same label with some other documents, can be ranked at multiple positions without changing ranking accuracy, i.e. x can have multiple true rank positions $p(x)$. Therefore, it is necessary

[1] True rank position $p(x)$ is the position of example x if we rank all examples under the same query by their ground truth.

to find a precise method to map the relevance labels into rank positions. A general method is to utilize labels to estimate the probability that one document is ranked at each position under the given query, such that all the documents with the same label have equal probability to be ranked at the same position, and those with better relevance labels have higher probability to be ranked at higher positions.

There are many specific ways for implementing this general method. In this paper, we introduce one based on the *equivalent correct permutation set*. Given a query q, assume all the documents under q are labeled using a k-level relevance judgment; and for each label level $t, (t \in \{0, 1, \cdots, k-1\})$, assume there are n_t documents under q whose labels are t. For the document list under q, we define an *equivalent correct permutation set* S:

$$S = \{\tau | g(x_i) > g(x_j) \Rightarrow \tau(x_i) < \tau(x_j)\}, \tag{7}$$

which means, for each permutation $\tau \in S$, if the relevance label of one document x_i is better than another document x_j, i.e. $g(x_i) > g(x_j)$, then the position of x_i in τ is higher than that of x_j, i.e., $\tau(x_i) < \tau(x_j)$. Then, the probability that a document x with label t is ranked at certain position ρ can be defined as:

$$P(p(x) = \rho) = \frac{1}{|S|} \sum_{\tau \in S} \mathbf{1}_{\{p(x)=\rho \text{ in } \tau\}}, \tag{8}$$

where $\mathbf{1}_{\{p(d)=\rho \text{ in } \tau\}}$ is an indicator function which equals 1 if document x is at position ρ in permutation τ and otherwise 0. Then, the probability can be calculated as:

$$P(p(x) = \rho) = \begin{cases} \frac{1}{n_t} & , \quad 1 + \sum_{m=t+1}^{k-1} n_m \leq \rho \leq \sum_{m=t}^{k-1} n_m \\ 0 & , \quad \text{otherwise} \end{cases}$$

For example, assume under a query q, there are five documents $\{a, b, c, d, e\}$. A three-level labeling is used. Assume the label set is $\{0, 1, 2\}$ where 2 means highest relevance. Assume the labels of five documents are $\{2, 2, 1, 1, 0\}$ respectively. Based on the above method, both a and b have probability 50% to be ranked at position 1

and 2, and 0 at other positions; both c and b have probability 50% to be ranked at position 3 and 4, and 0 at other positions; e has probability 100% to be ranked at position 5.

3.3.2 Learning Methods

To learn the ranking function, we need to minimize the query-dependent loss function with respect to the ranking parameters, denoted as ω. We use f_ω to represent the ranking function defined by the parameter vector ω. To this end, the query dependent loss function can be defined as:

$$L_f = \sum_{q \in \mathcal{Q}} \alpha(q) L(f_\omega; q, \mathcal{C}_I) + \beta(q) L(f_\omega; q, \mathcal{C}_N) \tag{9}$$

The straightforward method is to first obtain pre-defined categorization for each query and then learn the parameters of ranking function with pre-defined query categorization. In particular, for pre-defined informational queries, $\alpha(q) = 1, \beta(q) = 0$; while for pre-defined navigational or transactional queries, $\alpha(q) = 0, \beta(q) = 1$. If there exists soft query categorization, i.e. $\alpha(q) + \beta(q) = 1$ and $\alpha(q), \beta(q) > 0$, we can also use them directly in the loss function.

According to Eq. 6, the position-sensitive query-dependent loss function can be represented as the sum of example-level loss. We assume the original loss function are convex. Since there is no new parameters for pre-defined categorization, and all the example-level loss are part of original loss function, the proposed query-dependent loss function are convex as well. Therefore, we can apply the gradient descent method with respect to parameters of the ranking function to minimize the query-dependent loss function.

• ***Unified*** **Method:**

Due to the fact that existing query categorization may not be best for ranking and that even this kind of knowledge may not be available at learning time, we propose

Algorithm 1: **Unified Learning Method**

input : a set of training examples for learning to rank;
queries defined by query features.

output: parameter vector of the ranking function, ω, and parameter vector of the query categorization, γ, which minimize the loss function L_f.

Algorithm:

Start with an initial guess, e.g. random values, for ω and γ;

begin

while $(L_f(\omega_k, \gamma_k) - L_f(\omega_{k+1}, \gamma_{k+1})) > \epsilon$ **do**

•***Step 1: Learning*** ω

using gradient descent to minimize L_f with respect to ranking function parameters ω, holding the query categorization parameters fixed, i.e.,

$$\omega_{k+1} \leftarrow \arg\min_{\omega} \sum_{q \in \mathcal{Q}} \alpha_{\gamma_k}(q) L(f_{\omega_k}; q, \mathcal{C}_I) + \beta_{\gamma_k}(q) L(f_{\omega_k}; q, \mathcal{C}_N)$$

•***Step 2: Learning*** γ

using gradient descent to minimize L_f with respect to query categorization parameters γ, holding the ranking function parameters fixed, i.e.,

$$\gamma_{k+1} \leftarrow \arg\min_{\gamma} \sum_{q \in \mathcal{Q}} \alpha_{\gamma_k}(q) L(f_{\omega_{k+1}}; q, \mathcal{C}_I) + \beta_{\gamma_k}(q) L(f_{\omega_{k+1}}; q, \mathcal{C}_N)$$

end

a new method learning the ranking function jointly with query categorization. We refer to this new method as **unified** method.

Considering that query categorization, as hidden information in learning, is defined by a set of query features, which are disjointed with features of the ranking function. We assume $\mathbf{z}_q$ is the feature vector of query q and γ is the vector of parameters of query categorization, and we use the logistic function to obtain the query categorization $\alpha_\gamma(q)$ and $\beta_\gamma(q)$ from query features:

$$\alpha_\gamma(q) = \frac{\exp(\langle \gamma, \mathbf{z}_q \rangle)}{1 + \exp(\langle \gamma, \mathbf{z}_q \rangle)}, \qquad \beta_\gamma(q) = \frac{1}{1 + \exp(\langle \gamma, \mathbf{z}_q \rangle)}, \tag{10}$$

where $\langle \cdot, \cdot \rangle$ denotes the usual inner product.

Therefore, the query-dependent loss function can be represented as:

$$L_f = \sum_{q \in \mathcal{Q}} \alpha_\gamma(q) L(f_\omega; q, \mathcal{C}_I) + \beta_\gamma(q) L(f_\omega; q, \mathcal{C}_N) \tag{11}$$

which contains both ranking parameters ω and query categorization parameters γ. We need to minimize with respect to both ω and γ. We introduce a new algorithm, as shown in Algorithm. 1, to alternates between minimizing the loss with respect to ω and γ. Similar to the straightforward learning method, we use gradient descent methods in each iteration.

Remark: Note that, since we do not need information of query categories during testing, γ will not be used for ranking during testing; but γ can be used to compute the query categorization of testing queries.

Then, let's look at an algorithmic property of the unified learning method before applying it to specific ranking models.

Theorem 1 *Algorithm 1 converges in a finite number of steps if the original loss function, from which the query-dependent loss function is derived, is convex and bounded below.*

Proof 1 *As discussed above, the query-dependent loss function is derived from an original loss function $L_0(f; q)$ of one particular ranking model. We assume the original loss function $L_0(f; q)$ is convex and bounded below, i.e., $L_0(f; q) \geq c > -\infty$.*

To prove the convergence of Algorithm 1, it is necessary to show that the algorithm can decrease the loss in each iteration and that the loss is bounded. First, in the step of Learning ω*, according to Eq. 6, $L(f_\omega; q, \mathcal{C})$ has the same form of example-level loss with L_0 but assigns different pre-defined weights for different examples according to query categorization. Thus, $L(f_\omega; q, \mathcal{C})$ is convex with respect to ω due to L_0 is convex. To this end, we can use gradient descent to minimize the loss function with respect to ω based on fixed γ. Similarly, in the step of* Learning γ*, since $\alpha_\gamma(q)$ and $\beta_\gamma(q)$ in Eq. 10 are convex with respect to γ, we can use gradient descent to minimize the*

loss with respect to γ based on fixed ω. Therefore, our algorithm can decrease the loss function in each iteration.

Then, due to the same form of example-level loss with L_0, $L(f_\omega; q, \mathcal{C})$ has the lower bound since L_0 is bounded. And, it is easy to verify that $\alpha_\gamma(q)$ and $\beta_\gamma(q)$ are bounded because $\alpha_\gamma(q), \beta_\gamma(q) > 0$ and $\alpha_\gamma(q)+\beta_\gamma(q) = 1$. Given that $\alpha_\gamma(q)$, $\beta_\gamma(q)$ and $L(f; q, \mathcal{C})$ are all bounded, it can be verified that the query-dependent loss function has the lower bound.

In a conclusion, given the fact that our algorithm can decrease the query-dependent loss function in each iteration and that the query-dependent loss function has the lower bound, we can prove that Algorithm 1 can converge in a finite number of steps. Suppose that under the initial guess of ω and γ, the loss function has the value of $\mathcal{L}$, since the decreasing of the loss function is more than ϵ in each iteration except the last one, the algorithm can converge using no more than $\frac{\mathcal{L}}{\epsilon}$ steps.

Currently, for many of existing popular ranking algorithms, such as RankNet [16], RankSVM [35], RankBoost [22], ListNet [19], and ListMLE [77], their original loss functions are convex and bounded below. Therefore, we can apply our proposed query-dependent loss function to these ranking algorithms. In the following of this section, we will apply the position-sensitive query-dependent loss function to two particular ranking algorithms, RankNet [16] and ListMLE [77].

3.3.3 Example Query-Dependent Loss Functions

3.3.3.1 Example I: Query-Dependent Loss Functions for RankNet

RankNet [16] uses a loss function that depends on the difference of the outputs of pairs of training samples $x_i \succ x_j$ which indicates x_i should be ranked higher than x_j. The loss function is minimized when the document x_i with a higher relevance label receives a higher score, i.e., when $f(x_i) > f(x_j)$.

Let P_{ij} denote the probability $P(x_i \succ x_j)$, and let $\bar{P}_{ij}$ denote the desired target

values. Define $o_i \equiv f(x_i)$ and $o_{ij} \equiv f(x_i) - f(x_j)$. RankNet uses the cross entropy loss function [16]:

$$L(o_{ij}) = -\bar{P}_{ij} \log P_{ij} - (1 - \bar{P}_{ij}) \log(1 - P_{ij}),$$

where the map from outputs to probabilities are modeled using a logistic function: $P_{ij} \equiv e^{o_{ij}}/(1 + e^{o_{ij}})$. Then the final cost becomes: $L(o_{ij}) = -\bar{P}_{ij} o_{ij} + \log(1 + e^{o_{ij}})$.

For a pair of documents, $\langle x_i, x_j \rangle$, assume $p(i)$ and $p(j)$ are true ranking positions for x_i and x_j, respectively; $L(o_{ij})$ will be added into the loss function for navigational and transactional queries if only $p(i) \in \Phi(q, \mathcal{C}_N)$ or $p(j) \in \Phi(q, \mathcal{C}_N)$; Similarly, $L(o_{ij})$ will be added into the total loss for informational queries if only $p(i) \in \Phi(q, \mathcal{C}_I)$ or $p(j) \in \Phi(q, \mathcal{C}_I)$. To this end, the position-sensitive query-dependent loss function of RankNet for each pair can be defined as:

$$\begin{aligned} L(o_{ij}, q) &= \sum_{p(i)=1}^{n_q} P(p(i)|x_i, g(x_i))(\alpha(q) \cdot \mathbf{1}_{\{p(i) \in \Phi(q, \mathcal{C}_I)\}} \\ &\quad + \beta(q) \cdot \mathbf{1}_{\{p(i) \in \Phi(q, \mathcal{C}_N)\}}) \cdot L(o_{ij}), \end{aligned}$$

where n_q is the number of associated documents for query q; $P(p(i)|x_i, g(x_i))$ is the probability that x_i with label $g(x_i)$ is ranked at position $p(i)$, which is calculated using the method in Section 3.3.1.1.

3.3.3.2 Example II: Query-Dependent Loss Functions for ListMLE

ListMLE [77] learns a ranking function by taking individual lists as instances and minimizing a loss function defined on the predicted list and the truth list. In particular, ListMLE formalizes learning to rank as a problem of minimizing the likelihood function of a probability model. ListMLE seeks to minimize top-k surrogate likelihood loss [77], which is defined as:

$$L(f; q) = \phi(\Pi_f(\mathbf{x}), \mathbf{y}) = -\log P_{\mathbf{y}}^k(\Pi_f(\mathbf{x}))$$

where $\mathbf{x} = \{x_1, \cdots, x_n\}$ is the list of documents, and n is the number of associated document for query q; $\mathbf{y} = \{y(1), \cdots, y(n)\}$ is the true permutation of documents

under q, and $y(i)$ denotes the index of document which is ranked at position i; ϕ is a surrogate loss function; $\Pi_f(\mathbf{x}) = \{f(x_1), \cdots, f(x_n)\}$ denotes the permutation ordered by ranking function f; and $P^k_{\mathbf{y}}(\Pi_f(\mathbf{x}))$ is defined as:

$$P^k_{\mathbf{y}}(\Pi_f(\mathbf{x})) = \prod_{j=1}^{k} \frac{\exp(f(x_{y(j)}))}{\sum_{t=j}^{n} \exp(f(x_{y(t)}))}$$

where k is the parameter which infers that parameterized negative top-k log-likelihood with Plackeet-Luce model is used as the surrogate loss [19].

To build the position-sensitive query-dependent loss function, top-k_N surrogate likelihood loss is used for navigational or transactional queries, i.e. $\Phi(q, \mathcal{C}_N) = \{1, \cdots, k_N\}$; while top-$k_I$ surrogate likelihood loss is used for informational queries, i.e. $\Phi(q, \mathcal{C}_I) = \{1, \cdots, k_I\}$. To this end, the query-dependent loss function of ListMLE for each query can be defined as:

$$L(f; q) = -\alpha_q \log P^{k_I}_{\mathbf{y}}(\Pi_f(\mathbf{x})) - \beta_q \log P^{k_N}_{\mathbf{y}}(\Pi_f(\mathbf{x})) \tag{12}$$

Note that ListMLE has integrated rank positions into its loss function, we do not need to additionally estimate rank positions from relevance labels.

To learn the ranking functions using query-dependent loss functions for RankNet and ListMLE, we employ both the straightforward method with pre-defined query categorization and our proposed *unified* learning methods. In the straightforward method, we use the gradient descent method, which has been used to learn original RankNet [16] and ListMLE [77], to minimize the query-dependent loss function. In our proposed *unified* learning methods, we apply Algorithm 1 to both RankNet and ListMLE. The computation details of this method on RankNet and ListMLE, though tedious, are rather standard and will not be presented here.

3.4 Experiments

3.4.1 Experimental Setup

In our experimental study, we use the publicly available LETOR 3.0 data set [47] which is a benchmark data set for research on learning to Rank. We use TREC2003 and TREC2004 in LETOR 3.0 to evaluate the performance of learning to rank using query-dependent loss functions. Both of these two tracks categorize all the queries into three search tasks: topic distillation, homepage finding and named page finding. According to their characteristics, we view topic distillation queries as informational queries while homepage finding and named page finding queries as navigational or transactional queries. The statistics of the queries, feature definitions and relevance judgments for the Letor data set can be found in [47].

To define the features of queries (i.e., the $\mathbf{z}_q$ vector used in Eq. 10), we simply follow the heuristic method proposed in [24]. For each query q, we use a reference model (BM25 in this paper) to find its top-T ranked documents, and take the mean of the feature values of the T documents as the features of the query. In our work, we set $T = 50$.

To evaluate the performance of query-dependent loss functions for learning to rank, we compare the ranking methods as shown below.

- **RankNet:** In this method, we apply RankNet [16], which uses original query-independent loss function for ranking. This method has been showed in [16] to have good performance in ranking.

- **SQD-RankNet:** In this method, we apply query-dependent loss function for RankNet, with pre-defined query categorization. And, we adapt the straightforward learning method, as mentioned in section 3.3.2, to learn the ranking function. We denote this method as a *simple* query-dependent (SQD) method.

- **UQD-RankNet:** In this method, we apply query-dependent loss function for

RankNet, without query categorization. And, we adapt the *unified* learning method, as proposed in Alg 1, to learn the ranking function as well as query categorization simultaneously.

- **ListMLE:** In this method, we apply ListMLE [77], which uses original query-independent loss function for ranking. This method has been shown in [77] to have better performance even than many state-of-the-art ranking methods.

- **SQD-ListMLE:** In this method, we apply query-de- pendent loss function for ListMLE, with pre-defined query categorization. And, we adapt the straightforward method, as mentioned in section 3.3.2, to learn the ranking function. We also denote it as a *simple* query-dependent (SQD) method.

- **UQD-ListMLE:** In this method, we apply query-dependent loss function for ListMLE, without query categorization. And, we adapt the *unified* method, as proposed in Alg 1, to learn the ranking function as well as query categorization simultaneously.

In the experiments, we adapt two IR metrics, Normalized Discounted Cumulative Gain (NDCG) [31] and Mean Average Precision (MAP), to evaluate the performance of the learned ranking functions.

For a ranked list of documents, the NDCG score [31] at position n is calculated as follows:

$$\text{NDCG}(n) \equiv Z_n \sum_{j=1}^{n} \begin{cases} 2^{r(j)} - 1 & , \quad j = 1 \\ \frac{2^{r(j)}-1}{\log(j)} & , \quad j > 1 \end{cases} \tag{13}$$

where j is the position in the document list, $r(j)$ is the rating of the j-th document in the list (we represent the rating of *relevant* and *irrelevant* as 1 and 0 respectively in this paper), and Z_n is the normalization factor which is chosen so that the perfect list gets a NDCG score of 1.

MAP is calculated as:

$$MAP = \frac{1}{|Qr|} \sum_{q \in Qr} \frac{\sum_{n=1}^{N} (P@n \times rel(n))}{|R_q|}$$

where Qr is a set of test queries, R_q is the set of relevant document for q, n is the rank position, N is the number of retrieved documents, $rel()$ is a binary function on the relevance of a given rank.

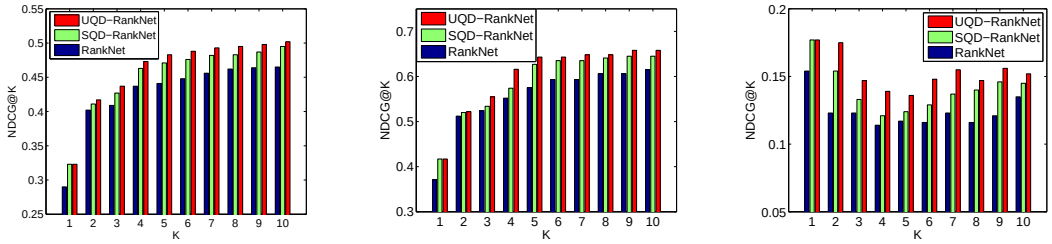
3.4.2 Experimental Results

We evaluate the performance of the ranking methods on TREC2003 and TREC2004 in LETOR. For each of these two data set, we conduct 5-fold cross-validation experiments. To ensure both navigational and informational queries have the same distribution in the 5 folds, we split navigational and informational queries into 5 folds, respectively. In SQD-RankNet, UQD-RankNet, SQD-ListMLE, and UQD-ListMLE, we set $k_N = 1$ and $k_I = 10$, i.e., $\Phi(q, \mathcal{C}_N) = \{1\}$ and $\Phi(q, \mathcal{C}_I) = \{1, \cdots, 10\}$.

3.4.2.1 Performance of query-dependent RankNet

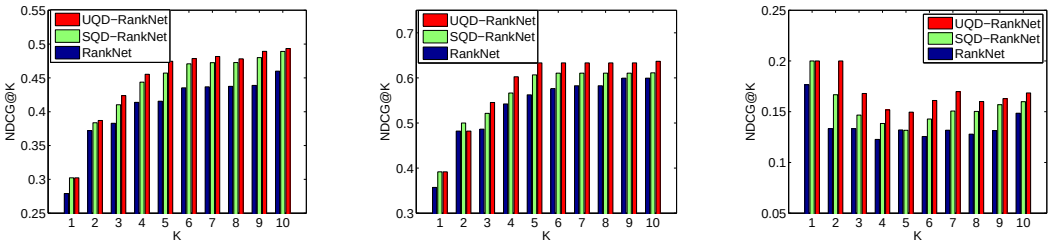
Figure 1 and 2 illustrate the NDCG values of both UQD-RankNet and SQD-RankNet compared with RankNet on TREC2003 and TREC2004, respectively. From Figure 1(a) and 2(a), we observe that these two query-dependent RankNet methods outperform the original RankNet on both data sets, and UQD-RankNet achieves better performance than SQD-RankNet. We also conduct t-test on the improvements in terms of mean NDCG, and the results indicate that for both data sets, the improvements of UQD-RankNet and SQD-RankNet over RankNet are statistically significant (p-value$<$ 0.04).

We also test the performance of ranking methods on the respective query types. From Figure 1(b), 1(c), 2(b), and 2(c), we observe that, on both TREC2003 and TREC2004 data sets, UQD-RankNet and SQD-RankNet give higher value on NDCG



(a) All query types (b) Navigational queries (c) Informational queries

Figure 1: Accuracy in terms of NDCG of query-dependent RankNet compared with original RankNet on TREC2003



(a) All query types (b) Navigational queries (c) Informational queries

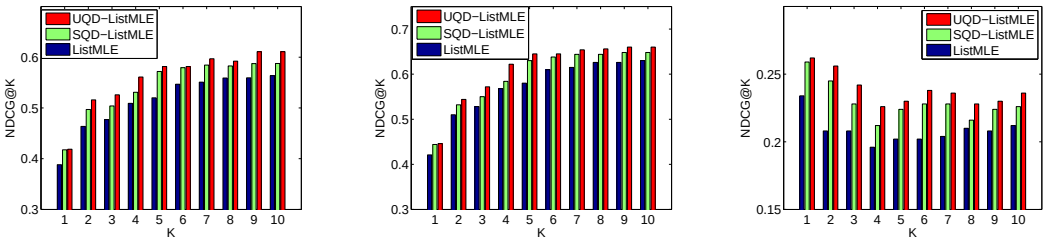
Figure 2: Accuracy in terms of NDCG of query-dependent RankNet compared with original RankNet on TREC2004

than RankNet for navigational and informational queries. In particular, for navigational queries, UQD-RankNet and SQD-RankNet perform better especially on NDCG@1 than that exhibited by RankNet; and for informational queries, NDCG@1 $\sim$ 10 gained by UQD-RankNet and SQD-RankNet are much higher than those exhibited by original RankNet. Moreover, UQD-RankNet outperforms SQD-RankNet for both navigational and informational queries.

In Table 1, we demonstrate the MAP value of UQD-RankNet and SQD-RankNet compared with RankNet on TREC2003 and TREC2004, respectively. From these two tables, we can see that query-dependent RankNet methods perform better than original RankNet, and UQD-RankNet reaches much better performance than SQD-RankNet. And, the t-test result shows that the improvement are statistically significant (p-value ¡ 0.05).

Table 1: MAP value of RankNet, SQD-RankNet, and UQD-RankNet

TREC2003			
	All Queries	Navigational	Informational
RankNet	0.372	0.501	0.135
SQD-RankNet	0.381	0.518	0.138
UQD-RankNet	**0.386**	**0.525**	**0.143**
TREC2004			
	All Queries	Navigational	Informational
RankNet	0.365	0.495	0.139
SQD-RankNet	0.375	0.511	0.142
UQD-RankNet	**0.381**	**0.517**	**0.149**



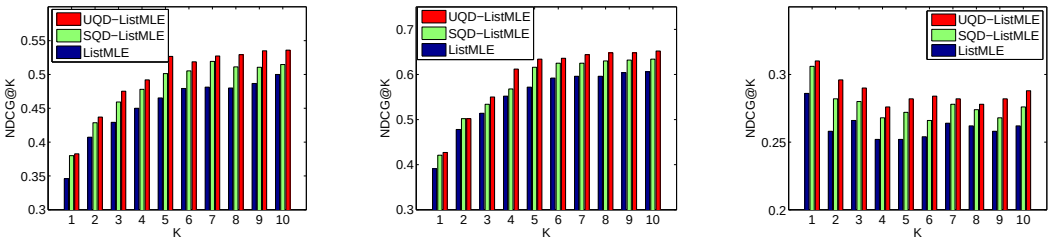
(a) All query types (b) Navigational queries (c) Informational queries

Figure 3: Accuracy in terms of NDCG of query-dependent ListMLE compared with original ListMLE on TREC2003

3.4.2.2 Performance of query-dependent ListMLE

Figure 3 and 4 demonstrate the NDCG values for both UQD-ListMLE and SQD-ListMLE compared with ListMLE on TREC2003 and TREC2004, respectively. From Figure 3(a) and 4(a), we observe that these two query-dependent ListMLE methods outperform the original ListMLE on both data sets, furthermore, UQD-ListMLE gives better performance than SQD-ListMLE. After conducting t-test in terms of mean NDCG, we find that, for both data sets, the improvements of UQD-ListMLE and SQD-ListMLE over ListMLE are statistically significant (p-value$<$ 0.04).

We also test the performance of ranking methods on respective query types. From the Figure 3(b), 3(c), 4(b), and 4(c), we can see that, on both TREC2003 and TREC2004 dataset, UQD-ListMLE and SQD-ListMLE give higher value on NDCG than ListMLE for navigational and informational queries. In particular,



(a) All query types (b) Navigational queries (c) Informational queries

Figure 4: Accuracy in terms of NDCG of query-dependent ListMLE compared with original ListMLE on TREC2004

for navigational queries, UQD-ListMLE and SQD-ListMLE performance better especially on NDCG@1 than that exhibited by ListMLE; and for informational queries, NDCG@1 $\sim$ 10 gained by UQD-ListMLE and SQD-ListMLE are much higher than those exhibited by original RankNet. Moreover, UQD-ListMLE outperforms SQD-ListMLE for both navigational and informational queries.

In Table 2, we illustrate the MAP value of UQD-ListMLE and SQD-ListMLE compared with ListMLE on TREC2003 and TREC2004, respectively. From these two tables, we can see that query-dependent ListMLE methods perform better than original ListMLE, and UQD-ListMLE reaches much better performance than SQD-ListMLE. And, we conduct the t-test whose result prove that the improvement are statistically significant (p-value ¡ 0.05).

Table 2: MAP value of ListMLE, SQD-ListMLE, and UQD-ListMLE

TREC2003			
	All Queries	Navigational	Informational
ListMLE	0.398	0.512	0.182
SQD-ListMLE	0.410	0.528	0.198
UQD-ListMLE	**0.418**	**0.537**	**0.210**
TREC2004			
	All Queries	Navigational	Informational
ListMLE	0.388	0.509	0.292
SQD-ListMLE	0.402	0.522	0.318
UQD-ListMLE	**0.410**	**0.530**	**0.334**

From these results, we can also find that ListMLE and query-dependent ListMLE

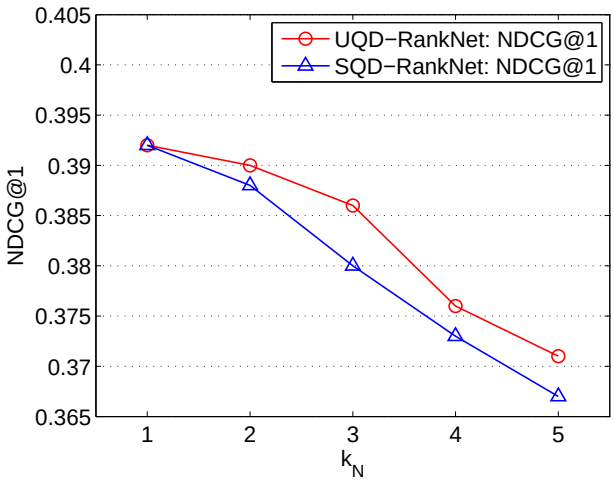


Figure 5: Ranking performance (NDCG@1) of UQD-RankNet and SQD-RankNet on navigational queries of TREC2004 against varying k_N

outperform RankNet and query-dependent RankNet, respectively, which illustrates that listwise ranking models can achieve better performance than pairwise ranking models.

3.4.2.3 Effects of Different Parameter Settings

In this experiment, we explore the effects of different settings of parameters k_N and k_I for ranking and perform comparison study by varying the value of k_N or k_I. We will illustrate the results performed on RankNet, and the experiment on ListMLE shows the similar results.

Figure 5 illustrates the performance of UQD-RankNet and SQD-RankNet on navigational queries of TREC2004 against varying the value of k_N. From the figure, we can find that increasing k_N can result in the decreasing performance of ranking for navigational queries.

Figure 6 demonstrates the performance of UQD-RankNet on informational queries on TREC2004 against varying the value of k_I. We can observe that, the accuracy of top-one position, in terms of NDCG@1, remains stable for varying k_I; too big or too small k_I can both cause the decreasing accuracy of other rank positions for

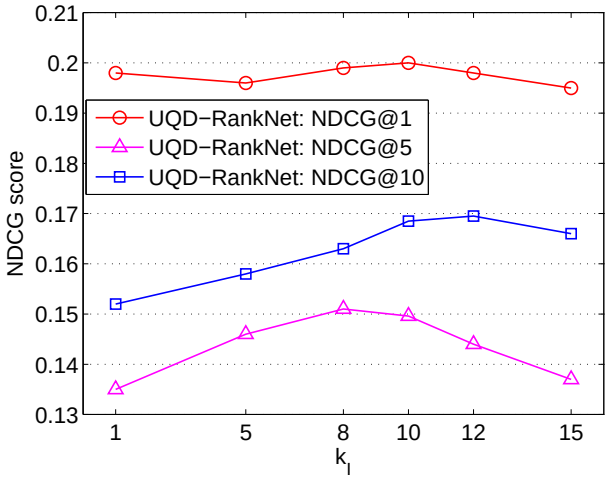


Figure 6: Ranking performance (NDCG@1,5,10) of UQD-RankNet on informational queries of TREC2004 against varying k_I

informational queries.

3.4.3 Discussions

3.4.3.1 Availability of Query-Specific Information in Training and Testing

As presented in Section 3.2.2, we use explicit query categories (straightforward learning method) or query-specific features (*unified* learning method) to learn the ranking models with query-dependent loss functions. However, when we employ the learned ranking model to perform ranking on new queries, we do not use any information of query classes or query-specific features for the new query. We hypothesize the reason that ranking models using query-dependent loss functions can outperform the original ranking models even without using query-specific information at *query time* as follows: Although query-specific classes and features are not available at query time, they can be viewed as extra tasks for the learner. Therefore, these query-specific information of training data set are transferred into other common features as training signals. We can benefit from ranking models with query-dependent loss functions due to the information in the extra training signals serving as a query-specific inductive bias for ranking [20].

3.4.3.2 *Query Categorization in Unified Learning Methods*

The above experimental results illustrate that it is possible to achieve better ranking performance by using the *unified* learning method than the straightforward method. We give two possible explanations. First, the straightforward method uses hard query categorization in learning, however, some queries may not be purely navigational or informational; therefore, the *unified* method using soft query categorization can define the more precise loss function for each query. Second, the pre-define query categorization may not be best for ranking. The *unified* method learns query categorization in tandem with the optimization of the ranking function, such that the obtained query categorization, even if contradicted with pre-defined categorization, can give more contribution to building precise ranking models than the straightforward method. In the following, we will demonstrate two specific examples on these two explanations.

Table 3 and 4 show the top ten results (Doc IDs) of RankNet, SQD-RankNet and UQD-RankNet for the query with ID 87 and those with ID 52, respectively. The Doc IDs of the relevant documents for these two queries are bold and red colored.

Table 3: Top 10 Results (Doc IDs) of One Informational Query (Query ID: 87)

RankNet	SQD-RankNet	UQD-RankNet
16116	61542	61542
151538	151538	**183864**
139846	156752	16116
61542	139846	156986
183864	**183864**	**165686**
156752	141943	**397632**
141943	**165686**	462230
33723	33723	**156335**
158984	158984	151538
165686	**156335**	144691

Query 87 is categorized as informational in TREC2004. Thus, it is 100% of informational and 0% of navigational in SQD-RankNet. However, learning using

UQD-RankNet renders this query about 63% informational and about 37% navigational. From the table 3, we can find that UQD-RankNet outperforms SQD-RankNet and RankNet for this query. In particular, UQD-RankNet and SQD-RankNet boost more relevant results into the top 10 rank positions by using query-dependent loss functions; Furthermore, UQD-RankNet boost relevant results to higher rank positions than SQD-RankNet. It illustrates that we can boost the accuracy of ranking by incorporating soft query categorization in learning to rank. Note that such soft categorization is learned jointly with learning ranking function.

Table 4: Top 10 Results (Doc IDs) of One Navigational Query (Query ID: 52)

RankNet	SQD-RankNet	UQD-RankNet
2003	13701	**2003**
13701	**2003**	**93720**
93720	124621	13701
542787	220784	**68046**
178575	14266	**60571**
39599	**93720**	124621
68046	149052	220784
124621	59495	8889
8889	**68046**	14266
60571	542787	149052

Query 52 is categorized as navigational in TREC2004. Thus, it is 0% of informational and 100% of navigational in SQD-RankNet. However, after learning using UQD-RankNet, this query is about 58% to be informational and about 42% to be navigational, which means UQD-RankNet consider this query more like an informational query. From the table 4, we can find that UQD-RankNet outperforms SQD-RankNet and RankNet for this query. Moreover, even RankNet outperforms SQD-RankNet for this query, which indicates that it may decrease the accuracy of ranking by considering this query as totally navigational. This example demonstrates that what is good in terms of query categorization is not necessarily good in terms of ranking, and we can boost the accuracy of ranking by using query categorization learned jointly with learning ranking function.

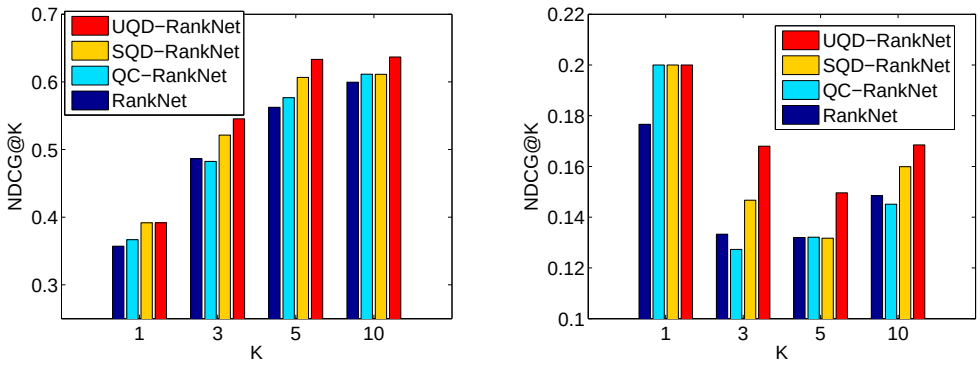
3.5 Query-Dependent Loss Functions v.s. Query-Dependent Rank Functions

The importance of query-dependent ranking is now widely recognized. Several previous studies have exploited another query-dependent method: query-dependent ranking functions. The key idea of those efforts is to employ different ranking functions for different queries. Kang et al. [38] classified queries into two categories based on search intentions and built two different ranking models accordingly. Geng et al. [24] proposed a K-Nearest Neighbor (KNN) method to employ different ranking models for different queries. In addition, Zha et al. [79] tried another query-dependent method, which implicitly incorporates query differences using monotone transformations of the learned ranking functions.

In this section, we compare the performance between the ranking function using query-dependent loss function and query-dependent ranking function. We employ RankNet to construct individual ranking functions for navigational queries and informational queries, respectively. We denote this approach as QC-RankNet.

Figure 7 illustrates the performance of UQD-RankNet, SQD-RankNet, QC-RankNet and RankNet on navigational and informational queries of TREC2004, respectively. Notice that QC-RankNet for navigational queries is trained using only the navigational queries with associated documents from the training set, and so is QC-RankNet for the informational queries; While the other three methods are learned using the whole training set. From these figures, we can find that ranking functions using query-dependent loss function (UQD- and SQD-RankNet) outperform query-dependent ranking function (QC-RankNet) and query-indep- endent ranking (RankNet). After conducting t-test on the improvements in terms of mean NDCG, we find that the improvements of UQD-RankNet and SQD-RankNet over QC-RankNet are statistically significant (p-value$<$ 0.05).

Why ranking functions learned using query-dependent loss functions can achieve



(a) Navigational queries

(b) Informational queries

Figure 7: Ranking performance (NDCG@K) of UQD-RankNet, SQD-RankNet, QC-RankNet and RankNet on Navigational/Informational queries of TREC2004

better performance than the corresponding query-dependent ranking functions as well as query-independent ranking functions? We hypothesize a couple of reasons as follows.

Firstly, query-dependent loss function contains more useful information for ranking than the loss for query-independent ranking function or query-dependent ranking function. For query-independent approaches, we use the same loss function for all the query categories, i.e., we minimize $L(Q_I) + L(Q_N)$, where Q_I and Q_N represent the training group of informational and navigational queries, respectively; for query-dependent ranking functions, we use respective loss functions for each query category, i.e., we minimize $L_I(Q_I)$ and $L_N(Q_N)$ separately; however, for our proposed approach, we combine both the loss function for informational queries and that for navigational queries together on each query, but they are weighted according to the query's soft categorization, i.e., we minimize $\sum_{q \in Q_I \bigcup Q_N} \alpha(q)L_I(q) + \beta(q)L_N(q)$. Thus, we incorporate more information into query-dependent loss functions than the other two approaches.

Secondly, due to many queries that can fit into more than one categories, the ranking function using query-dependent loss functions outperform query-dependent ranking functions. For example, with the query "*united nations*", the user may expect

to get the homepage of the United Nations (a navigational intention), or the user may be looking for recent news regarding a United Nations resolution (an informational intention). For this kind of queries, it is difficult to decide one ranking function corresponding to a single query category to rank documents for the query. However, our proposed approach does not rely on explicit query categorization at query time, and the learned ranking function can be used on a broad spectrum of queries.

For another reason, there exists a number of labeled documents which are not critical for ranking in the training set, such as those should not be ranked in top positions. However, if this kind of documents are very difficult to rank, they may have much influence on the training process and attenuate the effect from important documents. By using query-dependent loss, we can filter out these hard-to-rank and unimportant documents from computing the risk of ranking so as to build a more effective ranking model from a healthier training process.

Furthermore, for query-dependent ranking function, it uses only a part of training dataset to learn the ranking model for each query category. It may cause the declining accuracy due to the lack of enough training examples. However, the approach of query-dependent loss can avoid the reduction in the number of training examples.

In addition, the shorter training and testing time is another advantage of query-dependent loss approach over query-dependent ranking approach. The training time of query-dependent ranking approach could be quite large, because many models need to be trained; while only one model is trained for query-dependent loss approach. And, at testing time, query-dependent ranking approach need to spend additional time in online searching the model which is best for each test query; while query-dependent loss approach use only one model. Therefore, it is more time-consuming to use query-dependent ranking approach than to use query-dependent loss approach.

3.6 Summary

In this chapter, we have proposed to incorporate query difference into constructing ranking models by introducing query-dependent loss functions. Based on a popular query taxonomy of Web search, we exploit the position-sensitive query-dependent loss function for ranking and apply it on two concrete ranking algorithms. Moreover, beyond the straightforward method learning ranking model with pre-defined query categorization, we have devised a new learning method which can conduct learning of the ranking model jointly with that of query categorization. Experimental results illustrate that the proposed query-dependent loss functions can significantly improves the accuracy of the learned ranking functions, and our new learning method achieve better performance than the straightforward method.

Although this ranking approach based on query-dependent loss functions has been proved more effective, it requires precise definition of query difference in the loss function, which is very hard when there are more query categories. Moreover, in practical search engines, there are new requirement for the ranking model such as deep dive and incremental update on dedicated models. Motivated by all of these considerations, we propose a new divide-and-conquer ranking framework in the next chapter.

CHAPTER IV

RANKING SPECIALIZATION FOR WEB SEARCH

The ranking approach based on query-dependent loss functions requires precise definition of query difference in the loss function, which is very hard when there are more query categories. Furthermore, in practice, the commercial search engine usually requires deep dive and incremental update on dedicated models, which can improve the ranking performance on a specific subset of queries without hurting others. Motivated by these considerations, we propose a new divide-and-conquer framework for ranking specialization in this chapter.

As discussed in last chapter, various types of query difference make it difficult to build a single general ranking function for all kinds of queries, because the ranking function, while indicating good ranking relevance for a certain type of queries, may not be able to achieve similar performance for other types of queries. For instance, many current search engines can achieve good ranking performance on general short queries with 2-3 query terms, but they usually can not achieve high relevance for long queries with more query terms. And, for popular queries that are often searched by Web users, search engines can generally return good results, but for those that rarely occur or are new phrases on the Web, search engines may not be able to give sound ranking relevance. Before exploring new ranking methods to resolve these difficulties, we need to first investigate why query difference can result in such ranking problems.

We hypothesize the reason as the diverse feature impacts on ranking relevance with respect to different queries. For instance, for homepage finding (a kind of navigational) queries, the textual similarity between the query and the document title may be the best indicator of ranking relevance, whereas for topic distillation [76] (a

kind of informational) queries, the whole-document TFIDF and BM25 features may be better for inferring relevance. As another example, for popular queries, document popularity features, such as PageRank, may be important for ranking, whereas for rare queries, it is not necessary to use document popularity to measure the ranking relevance. As a conclusion, a single ranking model cannot reflect different feature impacts for different queries, such that it would not be adequate to use one single ranking model for diverse types of queries.

Therefore, it is necessary to find some new approaches which can give better ranking relevance for all the different types of queries. A straightforward method is to add query difference in terms of additional features into learning the single ranking function, however, since this method requires high quality of both the new features and training data, it usually does not effective in practice (as shown in experiment in Section 4.6.1.2). In this paper, we propose a divide-and-conquer framework for improving the ranking relevance for all queries.

In the following, we will introduce the new divide-and-conquer approach for ranking specialization for improving search relevance in details. We will start with introducing the general divide-and-conquer framework, followed by concrete discussion on three major parts of the framework in sequence.

4.1 A Divide-and-Conquer Framework

The divide-and-conquer framework consists of three major steps. In the first step, we target at identifying a set of ranking-sensitive query topics, $\mathcal{C}_{\text{query}} = \{C_1, C_2, \cdots, C_n\}$, based on all of training queries, $\mathcal{Q}_{\text{train}} = \{q_1, q_2, \cdots, q_N\}$. The recognized query topics are considered ranking-sensitive in the sense that different queries of the same topic should have similar characteristics in terms of ranking, especially, the similar family of important features for ranking. After this step, for each training query $q_i \in \mathcal{Q}_{\text{train}}$, we can obtain its distribution over all of extracted ranking-sensitive query topics, i.e.

Topic(q_i) = $\langle P(C_1|q_i), P(C_2|q_i), \cdots, P(C_n|q_i)\rangle$, where $P(C_i|q)$ is the probability that q belongs to C_i. This step is like *dividing* the problem of learning one single ranking model for all training queries into a set of sub-problems of learning the ranking model for each ranking-sensitive query topic. The number of query topics can be set either empirically to constants, or through gap statistic [75].

The second step is to develop a unified learning method for learning multiple ranking models M_k ($k = 1, 2, \cdots, n$), each exclusively corresponding to one ranking-sensitive query topic $C_k \in \mathcal{C}_{\text{query}}$. In our work, we propose a global loss function by combining risks of different ranking-sensitive query topics and introduce *Topical RankSVM* to train all the models $M_1, M_2, \cdots, M_n$, simultaneously, by minimizing the global loss function. By applying this unified learning method, we have considered dependency between different query topics when building their respective ranking functions, which can be beneficial to further improve ranking. Moreover, though treated unequally in learning each ranking function, all the training queries contribute to learn all ranking models of query topics, which avoids the lacking of training examples for learning the model of any single query topic. This unified method is quite general as we can use different feature set for different query topics. As incorporating information of query topics into the ranking algorithm, this step is like *conquering* the problem of learning the respective ranking model for each query topic.

The goal of the last step is to conduct ranking for new testing queries, $\mathcal{Q}_{\text{test}} = \{q_1, q_2, \cdots, q_t\}$. For each testing query $q_j \in \mathcal{Q}_{\text{test}}$, we apply an ensemble method, which try to minimize the risk consistent with the loss in training process. We first select a certain number H of ranking models M_{j1}, M_{j2},$\cdots$, M_{jH}, whose corresponding query topics hold H highest correlation with q_j, and then aggregating the ranking results S_{j1}, S_{j2},$\cdots$, S_{jH} obtained by M_{j1}, M_{j2},$\cdots$, M_{jH} into a final ranking results $\boldsymbol{S}$. After *divide-and-conquer*, this step aggregates ranking results from sub-models together into the improved final ranking results.

We summarize the general framework in Figure 8. Such ranking specialization approach allows us to use different feature sets or data sets to learn respective ranking models for different query topics, so as to boost the relevance for each query groups; And, the global loss in the second step serves as a unified relevance target when training different models for different query topics, such that we can optimize the overall search relevance when we train different ranking models. In the rest of this section, we will discuss the details of each step of the framework in sequence.

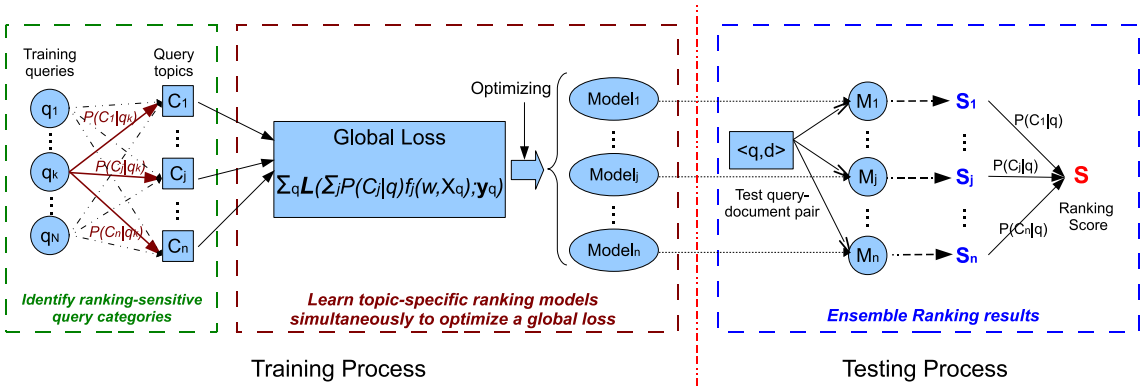


Figure 8: The ranking specialization framework for improving search relevance

4.2 Identifying Ranking-Sensitive Query Topics

4.2.1 Generating Query Features

To identify ranking-sensitive query topics, we first generate a set of features to represent queries by taking advantage of the ranking features of top pseudo feedbacks of the query. For each training query $q \in \mathcal{Q}_{\text{train}}$, we retrieve a set of pseudo feedbacks, $PF(q) = \{d_1, d_2, \cdots, d_T\}$, consisting of the top T documents ranked by a reference model (we use BM25 in this paper). The ranking features of query-document pair of $\langle q, d_i \rangle$ are defined as a feature vector $\mathbf{x}^{qd_i} = \langle x_1^{qd_i}, x_2^{qd_i}, \cdots, x_D^{qd_i} \rangle$ (D is the number of ranking features). To represent q in a feature space, we aggregate the ranking features of top-T pseudo feedbacks of q into a new feature vector. We take the *mean*

and *variance* of the ranking feature values as two aggregation methods. Thus, the feature vector of query q can be represented as

$$\langle \mu_1(q), \mu_2(q), \cdots, \mu_D(q), \sigma_1^2(q), \sigma_2^2(q), \cdots, \sigma_D^2(q) \rangle$$

where $\mu_k(q)$ denotes the mean value of k-th feature over q's pseudo feedbacks, and $\sigma_k^2(q)$ denotes the variance value of k-th feature over q's pseudo feedbacks.

In this paper, we will employ linear SVM model as the ranking algorithm. Thus, before generating query features, we have applied quantile normalization [13] on ranking features of query-document pairs in both training and testing dataset, such that the values of all ranking features are in the scale of $[0, 1]$. As a result, the values of extracted query features are also in the scale of $[0, 1]$.

4.2.2 Generating Query Topics and Computing Topic Distribution for Queries

After generating query features, we employ the mixture model as the clustering method to obtain ranking-sensitive query topics. We can either empirically set the number of cluster as a constant n, or determine it through gap statistic [75]. After learning the model, we can obtain a set of query clusters $\{C_1, C_2, \cdots, C_n\}$, where each cluster C_k can be represented as a vector $\mathbf{x}^{C_k} = \langle x_1^k, x_2^k, \cdots, x_{D_q}^k \rangle$, and D_q denotes the number of query features. We consider each cluster as one query topic.

Furthermore, we incorporate the prior knowledge of feature importance for ranking into identifying ranking-sensitive query topics. We obtain feature importance scores by using the ranking weights learned by a general RankSVM on a sample of training data. In our paper, we integrate the feature importance as weights into aggregated query features. Assuming the feature importance scores are $\mathbf{w} = \langle w_1, w_2, \cdots, w_{D_q} \rangle$, the weighted feature vector of query q_i is computed as

$$\mathbf{w} \odot \mathbf{x}^i \equiv \langle w_1 x_1^i, w_2 x_2^i, \cdots, w_{D_q} x_{D_q}^i \rangle \tag{14}$$

where $\mathbf{x}^i = \langle x_1^i, x_2^i, \cdots, x_{D_q}^i \rangle$ is the original feature vector of query q_i. Based on the new query features weighted by feature importance, we can employ the mixture model to obtain the ranking sensitive query topics with integrated feature importance.

Based on this representation of query topics, we are able to calculate the topic distribution Category$(q) = \{P(C_1|q), P(C_2|q), \cdots, P(C_n|q)\}$ for query q as

$$P(C_k|q) = \frac{|\mathbf{x}^q - \mathbf{x}^{C_k}|^2}{\sum_{i=1}^{n} |\mathbf{x}^q - \mathbf{x}^{C_i}|^2} \tag{15}$$

Since we take advantage of ranking features of top retrieved pseudo feedbacks of the query to generate query features as well as we compute topic distribution for queries by incorporating prior knowledge of feature importance for ranking, we consider the identified query topics and computed topic distribution for each query as ranking-sensitive.

4.3 A Unified Approach for Learning Multiple Ranking Models

4.3.1 Problem Statement

For traditional ranking approach, the task is to find a function f in the form of

$$y = f(X, \omega), \ f \in \mathcal{F}$$

where X denotes an $M \times D$ matrix representing D dimensional feature vectors of M documents; ω represents the unknown ranking parameters; y is a vector representing ranking scores of the M documents. The goal of learning is to select a best function $\hat{f}$, such that $\hat{f}$ minimizes the given loss function:

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} L(f(X_i, \omega), y_i) \tag{16}$$

where N is the number of queries in the training set; X_i denotes the set of documents associated with the i-th query; y_i is the vector of corresponding ranking scores; L denotes a defined query-level loss function. Clearly, traditional ranking approach learns single ranking function for all queries.

Inspired by the diverse ranking characteristics implied by different queries, to improve ranking relevance, we formalize a new problem of learning multiple ranking functions, $f_1, f_2, \cdots, f_n \in \mathcal{F}$, given the identified ranking-sensitive query topics $C_1, C_2, \cdots, C_n$, where each ranking model $f_i (i = 1, \cdots, n)$ can represent the ranking characteristics of its corresponding query topic C_i. In order to create a unified relevance target for all topic-specific ranking models and let all training examples contribute to all ranking models, we propose a new global loss function by combining ranking risks of all training examples with different weights according to the training query's similarity to different query topics. By optimizing this global loss function, we can learn multiple ranking functions, simultaneously. The intuition is that if the query of one training example is highly correlated to a certain query topic, this training example will contribute more to learn the ranking function of this query topic. Specifically, the global function is defined as

$$\langle \hat{f}_1, \cdots, \hat{f}_n \rangle = \arg \min_{f_1 \cdots f_n} \sum_{i=1}^{N} L(\sum_{j=1}^{n} P(C_j|q_i) f_j(X_i, \omega_j), y_i) \tag{17}$$

where n is the number of identified ranking-sensitive query topics; $P(C_j|q_i)$ represents the probability that q_i belongs to C_j; and ω_j denotes unknown parameters of the ranking function f_j corresponding to the query topic C_j. Intuitively, if query q_i is highly correlated to a query topic C_j, i.e. with high value of $P(C_j|q_i)$, the loss of ranking under q_i will be much associated with learning ω_j.

4.3.2 Topical RankSVM

The learning task in Eq. 17 is specified when the form of the ranking function f and that of the loss function L are defined, for example, we can use linear function as ranking function, i.e. $f(X, \omega) = \omega^T X$, and L_2 norm as loss function, i.e. $L(f(X, \omega), y) = \|f(X, \omega) - y\|^2$. The ranking specialization framework is quite general and flexible in the sense that it can apply different ranking algorithms. In this paper, we use RankSVM as an example to demonstrate its advantages. For simplicity,

we consider the linear function $f(X, \omega) = \omega^T X$. We refer to our method as *Topical RankSVM*. Note that the same idea can also be applied to other ranking algorithms, such as RankNet and RankBoost, by modifying the respective loss function according to Eq. 17.

• **RankSVM:**

We first make a review of RankSVM [26, 35]. Its learning task is defined as the following quadratic programming problem:

$$\begin{aligned} &\min_{\omega, \xi_{q,i,j}} \frac{1}{2}\|\omega\|^2 + c\sum_{q,i,j} \xi_{q,i,j} \\ &s.t.\ \omega^T X_i^q \geq \omega^T X_j^q + 1 - \xi_{q,i,j}, \\ &\forall X_i^q \succ X_j^q,\ \xi_{q,i,j} \geq 0 \end{aligned} \tag{18}$$

where $X_i^q \succ X_j^q$ implies that document i is ranked ahead of j with respect to query q in the training dataset; $\xi_{q,i,j}$ denotes slack variable; and $\|\omega\|^2$ represents structural loss.

• **Topical RankSVM:**

We then describe *Topical RankSVM*. Inspired by the fact that the ranking model of one specific ranking-sensitive query topic depends more on the training query-document pairs, the query in which has higher correlation to the certain query topic, we modify Eq. 18 into the optimization problem of *Topical RankSVM*:

$$\begin{aligned} &\min_{\omega, \xi_{q,i,j}} \frac{1}{2}\sum_{k=1}^{n} \|\omega_k\|^2 + c\sum_{q,i,j} \xi_{q,i,j} \\ &s.t.\ \sum_{k=1}^{n} P(C_k|q)\omega_k^T X_i^q \geq \sum_{k=1}^{n} P(C_k|q)\omega_k^T X_j^q + 1 - \xi_{q,i,j}, \\ &\forall X_i^q \succ X_j^q,\ \xi_{q,i,j} \geq 0 \end{aligned} \tag{19}$$

where ω_k denotes the parameters of ranking function with respect to query topic C_k. Note that we can use different feature sets for different query topics by using this method, but for simplicity, we didn't try it in this work. The optimization problem

can be solved by employing existing optimization techniques, the computation details of which, though tedious, are rather standard and will not be presented here.

Note that, there are several advantages by using *Topical RankSVM*. Firstly, we are able to embed ranking-sensitive query topics directly into the ranking algorithm and learn multiple ranking functions for different query topics, simultaneously. Secondly, the global loss serves as a unified relevance target for all topic-specific ranking models, which can boost the relevance than training models separately. And, we employ all of the training queries to learn ranking models of each query category, avoiding the reduction of the training examples. Moreover, compared to previous work [24, 38], our approach is not less efficient on training time, which has been proved in the experiments on a large scale dataset. The same idea can also be applied on ranking algorithms other than RankSVM, in the similar way. In the rest of this section, we will employ a testing method which is consistent with the training method, in the sense that both of them focus on optimizing the same risk of ranking.

4.4 Ensemble Ranking for New Queries

After obtaining multiple ranking models corresponding to ranking-sensitive query topics, we employ an un-supervised ensemble method for improving ranking at query time. The intuition is that: Similar queries in ranking-sensitive feature space are more likely to hold similar ranking characteristics, therefore, if one query topic has higher correlation to the testing query, the corresponding ranking models should contribute more to the final ranking of the testing query.

Assuming that $\hat{f}_1, \hat{f}_2, \cdots, \hat{f}_n$ are ranking models learned with the method in 4.3 and correspond to query topics C_1, C_2, $\cdots$, C_n respectively; $\tilde{q}$ is a testing query, and $\{\tilde{d}_1, \tilde{d}_2, \cdots, \tilde{d}_{M_{\tilde{q}}}\}$ is the set of documents to be ranked with respect to $\tilde{q}$; and $P(C_1|\tilde{q})$, $P(C_2|\tilde{q})$, $\cdots$, $P(C_n|\tilde{q})$ are the probabilities that the new testing query belongs to query topics. Then, we compute the ranking score $\mathcal{S}(\tilde{q}, \tilde{d}_i)$ for each document $\tilde{d}_i$ ($i =$

$1, \cdots, M_{\tilde{q}}$) as follows:

$$\mathcal{S}(\tilde{q}, \tilde{d}_i) = \sum_{k=1}^{n} P(C_k|\tilde{q}) \hat{f}_k(\mathbf{x}^{\tilde{q}\tilde{d}_i}, \omega_k) \tag{20}$$

where $\mathbf{x}^{\tilde{q}\tilde{d}_i}$ is the ranking feature vector of the query-document pair of $\tilde{q}$ and $\tilde{d}_i$; ω_k denotes parameters of $\hat{f}_k$. Then, we can obtain the final ranking of documents under $\tilde{q}$ according to the aggregated ranking scores computed by Eq. 20. Note that this testing approach tries to minimize the ranking risk consistent with that in training process.

4.5 Experimental Setup

This section presents our evaluation setup. First, we describe the datasets we used in the experiments (Section 4.5.1). Then, we describe our evaluation metrics (Section 4.5.2) and the ranking methods to compare (Section 4.5.3) for the experimental results reported in Section 4.6.

4.5.1 Data Collection

In the experiment, we used three datasets, including both the publicly benchmark dataset and that obtained from a commercial search engine.

LETOR 3.0:

LETOR 3.0 [47] is a benchmark dataset for research on ranking [1]. We use TREC2003 and TREC2004 in LETOR 3.0 to evaluate the performance of exploring ranking-sensitive query topics for improving ranking. TREC2003 contains 350 queries and TREC2004 contains 225 ones. For each query, there are about 1,000 associated documents. Each query-document pair is given a binary judgment: *relevant* or *irrelevant*. In total, there are 64 features for each query-document pair, which can be referred to [47] for the details.

[1] http://research.microsoft.com/en-us/um/beijing/projects/letor

Both of these two tracks classify all the queries into three pre-defined categories, including topic distillation (TD), homepage finding (HP) and named page finding (NP), according to search intent. The statistics of queries for three categories in LETOR 3.0 can be found in [47]. Note that, this pre-defined hard categorization can also be used to improve ranking by applying the same method in Section 4.3. In our experiment, we will compare the effects on improving ranking by using ranking-sensitive query topics with that by using the pre-defined categorization.

LETOR 4.0:

LETOR4.0 is a new release of benchmark dataset for research on ranking. It uses the web page collection (50M pages) and two query sets from Million Query track of TREC 2007 and TREC 2008, called MQ2007 and MQ2008 for short. There are about 1700 queries in MQ2007 with labeled documents and about 800 queries in MQ2008 with labeled documents. Each query-document pair is represented by a 46-dimensional feature vector. And we used the data with the setting of supervised ranking to evaluate the performance of our proposed ranking approach.

Commercial search engine dataset (SE-Dataset):

We also conduct experiment on a dataset obtained from a major commercial search engine. This dataset contains 71,810 training queries and 1,227,094 query-document pairs for training as well as 7,668 testing queries and 252,086 query-document pairs for testing. All the training and test queries are randomly sampled from the real user traffic to the search engine. Each query is associated with its retrieved documents, along with human judged labels that represent the degrees of relevance of those documents with respect to the queries. There are five levels of relevance: perfect, excellent, good, fair, and bad. Features for each query-document pair used in building the ranking functions can be roughly grouped into the following categories: text-matching features, link-based features, user-click features, query and page classification features. We denote this dataset as *SE-Dataset*.

This dataset classifies all the queries into four semantic domains, including autos domain ($\mathcal{D}_{\text{auto}}$), local domain ($\mathcal{D}_{\text{local}}$), product domain ($\mathcal{D}_{\text{product}}$), and travel domain ($\mathcal{D}_{\text{travel}}$). For each query, SE-Dataset provides the pre-computed similarity score between the query and each query domain. The value of the similarity score is in the range of $[0, 1]$, 0 meaning the smallest value of similarity while 1 meaning the largest value of similarity. There are some queries in SE-Data that have the 0 similarity scores with all of four pre-defined domains. In our experiment, we define a new soft query classification, which contains five classes, including $\mathcal{D}_{\text{auto}}$, $\mathcal{D}_{\text{local}}$, $\mathcal{D}_{\text{product}}$, $\mathcal{D}_{\text{travel}}$, and *general* domain $\mathcal{D}_{\text{general}}$. For each query q, we set the similarity score with respect to *general* domain class as 1, and after normalizing similarity scores with respect to all five classes, we can obtain a soft query classification.

4.5.2 Evaluation Metrics

In the experiments, we adapt two IR metrics, Normalized Discounted Cumulative Gain (NDCG) [31] and Mean Average Precision (MAP), to evaluate the performance of the learned ranking functions. The details of these two metrics have been introduced in Section 3.4.1.

4.5.3 Ranking Methods Compared

To evaluate the performance of our approach employing ranking-sensitive query topics in learning the ranking function, we compare the performance of the following methods:

1. RSVM: As a baseline method, we employ RankSVM (denoted as RSVM) and conduct training over all the training queries to learn one single ranking model. At testing time, we use the single model to generate ranking results for all testing queries.

2. CRSVM: In this method, we apply the pre-defined query categorization into our proposed SVM-based learning method, where each query category is viewed as one query topic. In LETOR 3.0 dataset, each query can only belong to only one category.

Thus, this method equals to separating the training queries based on query categorization and training a specialized model for each category of queries. In commercial search engine dataset, we can employ the pre-defined soft query categorization into our proposed SVM-based learning method directly. We call this method "*Semantic-class based RankSVM*", denoted as CRSVM. At testing time, for hard query categorization, we choose the ranking model according to the category of each testing query; for soft query categorization, we apply the ensemble approach in Section 4.4 to generate ranking results.

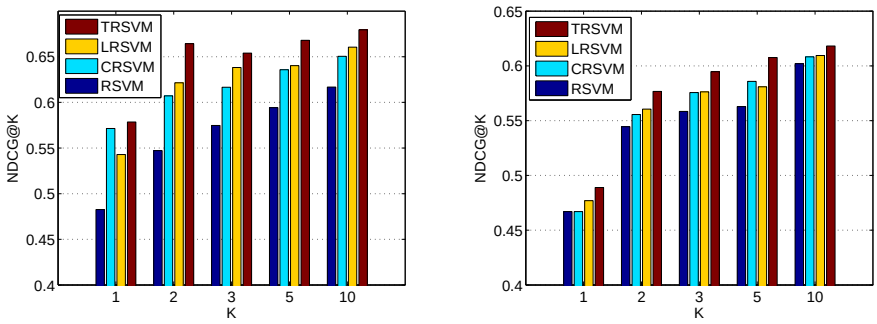
3. LRSVM: In this method, we simulate the general idea in [24]. After identifying the ranking-sensitive query categories, we classify each training query into the closest query category. Based on this hard partition of training queries, we train separate ranking model for each query category using its own fraction of training queries. This method is usually called "*Local Ranking*" in previous work. By using RankSVM, we denote this method as LRSVM. At testing time, according to the correlation between the test query and query categories, we select the ranking model of the most correlated query category and use it to generate the ranking results.

4. TRSVM: In this method, after identifying the ranking-sensitive query topics (Section 4.2), we employ the proposed "*Topical RankSVM*" (Section 4.3) with topic distribution of training queries to learn ranking models for those query topics. At testing time, we use the proposed ensemble method (Section 4.4) to generate the ranking results for testing queries.

4.6 Experimental Results

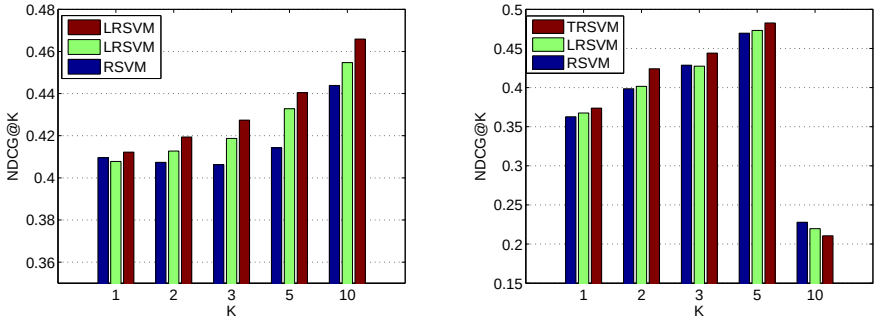
4.6.1 Experiments with LETOR Dataset

We evaluate the performance of the ranking methods on TREC2003 and TREC2004 in LETOR 3.0 as well as MQ2007 and MQ2008 in LETOR 4.0. For each of these datasets, we conduct 5-fold cross-validation experiments, using the default partitions



(a) TREC 2003 (b) TREC 2004

Figure 9: Ranking relevance in terms of NDCG of TRSVM compared with other methods on LETOR 3.0.



(a) MQ 2007 (b) MQ 2008

Figure 10: Ranking relevance in terms of NDCG of TRSVM compared with other methods on LETOR 4.0.

in LETOR. For TREC2003 and TREC2004, since the data of each class (TD, NP, HP) is split into 5 folds, we combine respective i-th fold ($i = 1, \cdots, 5$) of each class together to form up the i-th fold of the whole dataset.

To identify ranking-sensitive query topics, we use BM25 as the reference model to rank documents and choose the top $T = 50$ ranked documents (if the total number of documents under one query is lower than T, all documents will be used) as pseudo feedback to create ranking-sensitive features.

The topic number n is crucial to the performance of TRSVM and LRSVM. Since there are three pre-defined classes in TREC2003 and TREC2004, we select the value as $n = 3$ to compare the performance of TRSVM with that of LRSVM and CRSVM for experiments on TREC2003 and TREC2004. Since there is no pre-defined class

in MQ2007 and MQ2008, we will not test the performance of CRSVM on these two dataset. And, we set $n = 10$ to compare the performance of TRSVM with that of LRSVM for experiments on MQ2007 and MQ2008. In practice, this parameter is tuned automatically based on a validation set. In order to clearly illustrate the influence of this parameter on the ranking performance, we will also present the results with respect to different values of the topic/category number.

For RSVM, we can make use of its results provided in LETOR. For all the SVM models in the experiment, we employ the linear SVM. This is because the LETOR data set offers results of linear RankSVM.

4.6.1.1 Relevance Comparisons

The purpose of this experiment is to compare the average relevance of different ranking algorithms on different benchmark datasets. Figure 9(a) and 9(b) illustrate the NDCG values of TRSVM compared with RSVM, CRSVM and LRSVM on TREC2003 and TREC2004, respectively. From these two figures, we observe that, by building different ranking models with respect to different query categories/topics, TRSVM, CRSVM and LRSVM out-perform the single model learned by RSVM on both dataset. Furthermore, by extracting the ranking-sensitive query topics and applying the proposed unified learning method, TRSVM give better relevance than CRSVM and LRSVM. We conduct t-test on the improvements in terms of NDCG@3, and the results indicate that for both TREC2003 and TREC2004, the improvements of TRSVM over other ranking methods are statistically significant (p-value< 0.05).

In Table 5, we report the MAP scores of TRSVM compared with RSVM, CRSVM, LRSVM on TREC2003 and TREC2004, respectively. Table 5 indicates that TRSVM achieves much better relevance than RSVM, CRSVM and LRSVM. In particular, TRSVM achieves a gain of about 9% relative to RSVM on TREC2003 as well as 6% relative to RSVM on TREC2004; and TRSVM obtains more gains than CRSVM and

Table 5: MAP value of RSVM, CRSVM, LRSVM, and TRSVM on TREC2003 and TREC2004

Ranking Method	TREC2003	Gain	TREC2004	Gain
RSVM	0.578	-	0.501	-
CRSVM	0.601	+4%	0.513	+2%
LRSVM	0.605	+5%	0.521	+4%
TRSVM	0.628	+9%	0.532	+6%

LRSVM on both dataset.

Moreover, Figure 10(a) and 10(b) illustrate the NDCG values of TRSVM compared with RSVM, and LRSVM on MQ2007 and MQ2008, respectively. From these two figures, we also observe that, on both dataset TRSVM and LRSVM out-perform the single model learned by RSVM by employing different ranking models with respect to different query categories/topics. Furthermore, TRSVM, which extracts the ranking-sensitive query topics and applying the proposed unified learning method, can reach better ranking relevance than LRSVM based on local ranking. (Note that the NDCG@10 of queries in MQ2008 is much lower because a portion of queries have less than 10 documents to rank in this dataset.) We conduct t-test on the improvements in terms of NDCG@3, and find that the improvements of TRSVM over other ranking methods are statistically significant (p-value$<$ 0.05).

Table 6: MAP value of RSVM, LRSVM, and TRSVM on MQ2003 and MQ2004

Ranking Method	MQ2007	Gain	MQ2008	Gain
RSVM	0.464	-	0.470	-
LRSVM	0.474	+2%	0.477	+1%
TRSVM	0.481	+4%	0.489	+4%

Table 6 demonstrates the MAP value of TRSVM compared with RSVM as well as LRSVM on MQ2007 and MQ2008, respectively. From the table, we can observe that TRSVM achieves much better relevance than RSVM and LRSVM. In particular, TRSVM achieves a gain of about 4% relative to RSVM on both MQ2007 and MQ2008; and TRSVM obtains more gain than LRSVM on both datasets.

4.6.1.2 Results Analysis

In the following, we will investigate the reason that TRSVM can achieve better ranking relevance than the single model RSVM, the class-based ranking approach CRSVM, and the local ranking approach, LRSVM.

Table 7: Top 8 most important features for RSVM on TREC2003

RSVM
weighted in-link
weighted out-link
TF-IDF of title
TF of title
TF of body
TF-IDF of whole document
length of URL
topical PageRank

Table 8: Top 8 most important features for CRSVM on TREC2003

CRSVM (TD)	CRSVM (NP)	CRSVM (HP)
sitemap based score propagation	BM25 of whole document	number of slash in URL
sitemap based term propagation	LMIR.JM of whole document	HostRank
DL of title	sitemap based score propagation	HITS hub
length of URL	sitemap based term propagation	DL of URL
HostRank	LMIR.JM of anchor	length of URL
BM25 of title	BM25 of anchor	LMIR.JM of title
DL of URL	LMIR.DIR of whole document	sitemap based score propagation
BM25 of anchor	LMIR.ABS of whole document	sitemap based term propagation

Table 9: Top 8 most important features for TRSVM on TREC2003

TRSVM (topic-1)	TRSVM (topic-2)	TRSVM (topic-3)
sitemap based term propagation	number of slash in URL	length of URL
sitemap based score propagation	HostRank	outlink number
length of URL	HITS sub	sitemap based term propagation
number of slash in URL	sitemap based score propagation	sitemap based score propagation
DL or URL	sitemap based term propagation	number of slash in URL
weighted in-link	uniform out-link	HITS sub
number of child page	Outlink number	DL or URL
BM25 of title	LMIR.ABS of URL	DL or title

I. Multiple ranking models v.s. single model

Table 7, 8 and 9 demonstrates the 8 most important features for single model learned by RSVM and for separate models corresponding to query topics learned by CRSVM and TRSVM, respectively. These results are based on the experiment on TREC2003. The feature importance is measured by the absolute value of learned feature weight. The detailed description of features can be found in [47].

From these tables, we can observe that, introducing query difference in terms of query classification/clustering into ranking can help to build multiple ranking models with respect to various query classes (CRSVM) or query topics (TRSVM). These ranking models can represent multiple fine-grained ranking characteristics of various query classes/topics, while the single ranking model can only describe a coarse summarization of ranking characteristics over all various queries.

In particular, if we build one single ranking model using RSVM, the top 5 most important features of the model are *weighted in-link*, *weighted out-link*, *TF-IDF of title*, *TF of title*, and *TF of body*, as shown in Table 7 and 8. To verify whether these five features are most important to ranking for most of queries, we conduct an experiment as follows: we randomly sample 20 queries from the whole dataset and build respective ranking models for each query based on the documents and labels associated with the certain query, then we compute the respective feature importance of each query. We randomly sample 20 queries for 3 times, and there are only in average 6.3 queries (31.5%) whose top 5 most important features include at least three of top 5 most important features of the model learned by RSVM.

To gain an understanding of what is the better way to use the information of ranking-sensitive query topics to improve ranking, we perform a study on the effects of adding query topics information as features in learning. Specifically, for each query-document pair in both training and testing set, we use the query's topic distribution as additional features and apply RSVM to learn the single ranking model on expanded feature space. The testing results show that this method does not increase the ranking performance significantly, but even performs worse than the multiple ranking approach. The similar experiment on SE-Dataset also reports the same observation.

II. Ranking-sensitive query topics v.s. pre-defined semantic classes

After considering query different in terms of query classification(CRSVM)/clustering(TRSVM),

we can build multiple ranking models to represent different fine-grained ranking characteristics. For example, by using CRSVM, we can learn three ranking models corresponding to three query classes: topic distillation (TD), namepage finding (NP), and homepage finding (HP). Table 8 shows the top 8 most important features of each learned model. From this table, we can find that each of the three models learned by CRSVM are quite different with that of RSVM. In particular, some features, such as *TF-IDF of title*, *weighted out-link* etc., are important for RSVM, but are not essential for ranking of each specific query class. Furthermore, the most important features of each query class are diverse, for example, *sitemap based score/term propagation* are most important to TD queries; NP queries depend mostly on language model based (LMIR) and probabilistic (BM25) features; and *number of slash in URL* is the most important one for HP queries but not included in top 8 for the other two classes. Similarly, Table 9 shows that three ranking models learned by TRSVM are different with not only the single model of RSVM but also multiple models learned by CRSVM.

To verify whether TRSVM can obtain multiple ranking models that better represent ranking characteristics of respective query topics than other methods, we conduct the following experiment: for each identified query topic, we randomly sample 20 queries from those belonging to this query topic, and build respective ranking model for each of these 20 query based on their own associated documents and labels; then, we compute the respective feature importance of these queries and validate whether they include the top important features learned by TRSVM. We conduct this experiment on 5 query topics, and randomly sample 20 queries under each topic for 3 times. There are in average 14.8 queries (74%) whose top 5 most important features include at least three of those of the model learned by TRSVM. For LRSVM and CRSVM, the results shows that there are in average 12.7 (63.5%) and 11.3(56.5%) queries whose top 5 most important features include at least three of those of the model learned by LRSVM and CRSVM, respectively.

III. Why TRSVM performs better?

We hypothesize the reasons of why TRSVM can perform better than LRSVM and CRSVM as follows. Firstly, instead of pre-defined query classification, TRSVM and LRSVM represent each query by aggregating the ranking features of documents under such query and conduct un-supervised clustering method to identify ranking-sensitive query topics, which can be better to distinguish queries based on their various ranking characteristics.

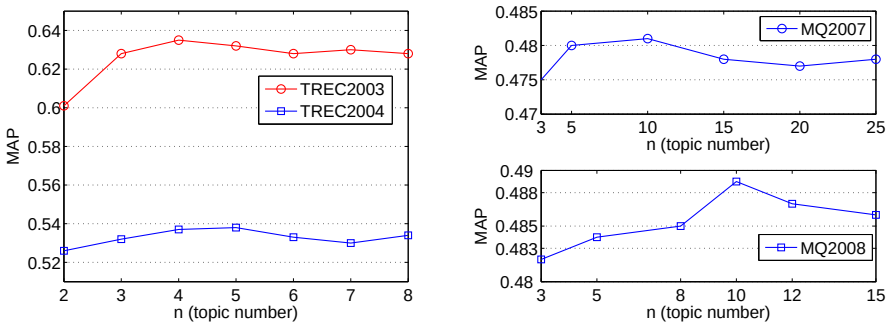
Secondly, LRSVM and CRSVM employ hard query classification/clustering and build multiple ranking models corresponding to each query class/cluster, however, many queries can fit into more than one classes/clusters, therefore, TRSVM can benefit ranking performance by taking advantage of soft query categorization into building multiple ranking models.

Another advantage of TRSVM is avoidance of the reduction in the number of training examples. Specifically, either LRSVM or CRSVM uses only a part of training dataset to learn the ranking model for each query class/cluster. It may cause the declining accuracy due to the lack of enough training examples. However, TRSVM can avoid the reduction of training dataset size since it uses all training examples, with different weights based on query soft clustering, in learning the ranking model of each query topic.

4.6.1.3 Effects of Different Parameter Settings

In this experiment, we explore the effects of different settings of the parameter n, i.e. the number of query topics, on ranking performance by conducting comparison study with varying the value of n.

Figure 11 show the performance of TRSVM on Letor dataset with varying values of n in terms of MAP. From the figure, we can find that, as n increases, the performances first increase, but then as n becomes much larger, there is no significant raising



(a) TREC2003 and TREC2004 (b) MQ2007 and MQ2008

Figure 11: Ranking performance (MAP) of TRSVM on the Letor datasets against varying topic number n

on the performance, and the performance is even deteriorated at some time. More specifically:

•When setting only a small number of query topics, the performances of TRSVM are not so good since the identified query topics are a bit broad to reflect the query difference.

•As setting the higher number of query topics, we can improve the performances since identified query topics are more fine-grained to reflect query difference in ranking.

•In the ideal case, with increasing number of query topics, the proposed method can achieve much better ranking performance, since we can build ranking models, each of which focuses on more fine-grained ranking characteristics. However, the actual results indicates that the ranking performances do not increase significantly but even deteriorate at some time. We hypothesize that, the extracted query features and identified ranking-sensitive query topics, though more effective than other query categorization method for enhancing ranking, are still not optimal for recognizing the ranking-sensitive query topics; therefore, when each identified query topic becomes

more fine-grained, the bias of query categorization will be enlarged so as to deteriorate the ranking performance.

4.6.2 Experiments with SE-Dataset

We compare the performance of proposed ranking methods (TRSVM) with the baselines of the single model approach (RSVM), class-based approach (CRSVM) and the local ranking based approach (LRSVM) on the commercial search engine dataset (SE-Dataset). The pre-defined classes in CRSVM include *auto*, *local*, *product*, *travel*, *general* as described in Section 4.5.1.

To identify ranking-sensitive query topics, we use BM25 as the reference model to rank documents and choose the top $T = 20$ documents (if the total number of documents under one query is lower than T, all documents will be used) as pseudo feedback to create ranking-sensitive features.

We set the query topic/cluster number in TRSVM/LRSVM, i.e. the parameter n, as $n = 20$ in the experiment. In practice, this parameter is tuned automatically based on a validation set. In order to clearly illustrate the influence of this parameter on the ranking performance, we will also present the results with respect to different values of the topic/category number.

4.6.2.1 Performance Comparisons

Figure 12 demonstrates the NDCG values of TRSVM compared with RSVM, CRSVM, and LRSVM on SE-Dataset. From the figures, we observe that, by building different ranking models with respect to different query categories/topics, TRSVM, LRSVM and CRSVM out-perform the single model learned by RSVM. Furthermore, by extracting the ranking-sensitive query topics and applying the proposed unified learning method, TRSVM give better performance than CRSVM and LRSVM. We conduct t-tests on the improvements in terms of NDCG@3, the results of which indicate that the improvements of TRSVM over RSVM, CRSVM and LRSVM are statistically

significant (p-value<0.05).

4.6.2.2 Effects of Different Aggregation for Query Features

In this experiment, we test the performance of the proposed TRSVM method with using different information for aggregating query features. By default, we compute the mean values of ranking features of top pseudo feedbacks as the feature vector of the query. In this experiment, we explore the effects of adding extra statistical quantities beyond means into the query feature vector. We use variance as the extra statistical quantity in our experiment. The ranking method using the expanded query features is denoted as TRSVM$_{var}$. Moreover, we test the effects of making use of the knowledge of ranking feature importance into the aggregation for query features. In particular, we first learn the ranking function by using a sample of the training dataset. After that, we can obtain the importance of each feature for learning the ranking function. Then, we incorporate feature importance as weight into computing query features. We denote the ranking method with incorporated feature importance as TRSVM$_{impt}$. We also test the ranking method which both expands query features with aggregated variance values and incorporates feature importance in identifying query topics, which is denoted as TRSVM$_{var-impt}$.

Figure 13 shows the performances of the proposed TRSVM method with applying various information into the aggregation for query features. From this figure, we observe that, utilizing feature importance into identifying query topics can increase the ranking performance. After conducting t-test on the improvement in terms of NDCG@3, we find that TRSVM$_{impt}$ out-perform TRSVM with p-value less than 0.02. Figure 13 also illustrates that expanding query features with aggregated variance value (TRSVM$_{var}$) does not improve ranking performance but even cause a decreased accuracy than TRSVM, the reason of which could be variance is not useful to identify ranking-sensitive query topics. However, it does not indicate that other

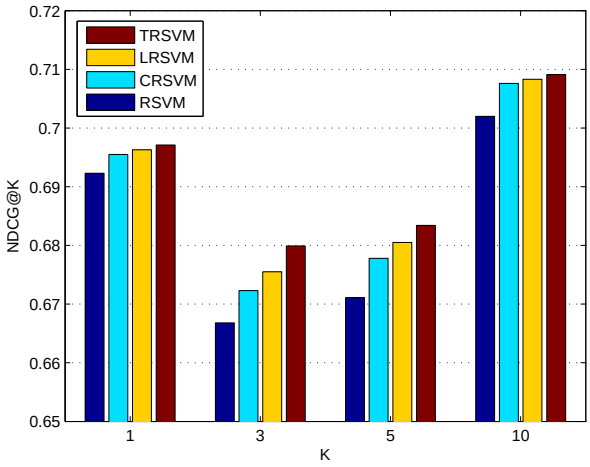


Figure 12: Relevance (ndcg@k) of TRSVM compared with the other methods on SE-Dataset

statistical quantities are not useful for query topic recognition either. It would be an interesting future work to explore what kind of statistical quantities are essential to identify ranking-sensitive query topics. Moreover, Figure 13 demonstrates that TRSVM$_{var-impt}$ can achieve better performance than TRSVM$_{var}$ and TRSVM, but not TRSVM$_{impt}$, which also indicate that using feature importance can boost ranking performance while expanding query features with aggregated variance may not be useful for improve ranking.

4.6.2.3 Robustness to Noisy Query Topics

In the following, we perform experiments to evaluate the robustness of our divide-and-conquer ranking approach to noisy ranking-sensitive query topics. We manually add some noises into the ranking-sensitive query topics and test this effects on the overall ranking relevance. Specifically, after identifying ranking-sensitive query topics, we randomly select a portion of training queries and change their topic distribution into random values. Then, we employ TRSVM with these noisy ranking-sensitive query topics.

Figure 14 illustrates the NDCG@1,3,5 scores for the testing queries (on MQ2007

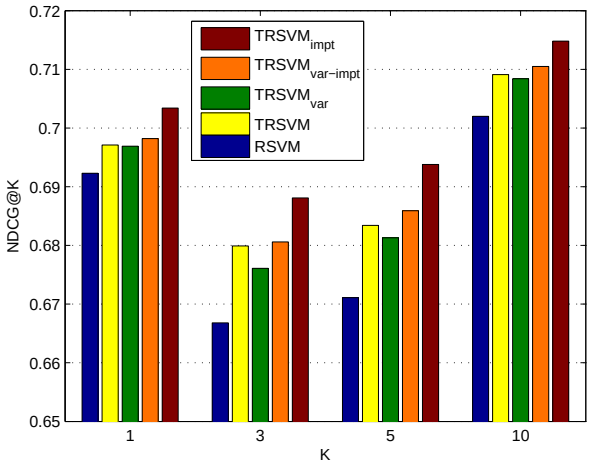


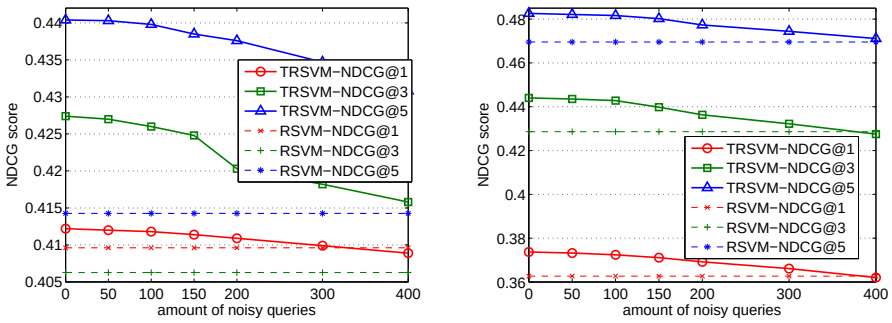
Figure 13: Relevance (ndcg@k) of TRSVM with different aggregation method for query features on SE-Dataset

and MQ2008) against varying amount of queries with noisy topic distribution in the training data. The figures show that, although the relevance declines as the amount of queries with noisy topic distribution increases, TRSVM even outperform single ranking model approach RSVM and therefore is robust to the noisy ranking-sensitive query topics. And we can find the similar results when we take the same experiments on LETOR 3.0 dataset.

The robustness of our divide-and-conquer ranking approach can also be demonstrated on SE-Dataset, especially from Figure 13. This figure shows that different aggregation methods give rise to different ranking-sensitive query topics, and hence different ranking performance; but, in spite of such difference, any of these different query categorizations can be used to trained the ranking models which outperform the single model approach (RSVM). Therefore, the proposed divide-and-conquer framework is robust to the noisy ranking-sensitive query topics.

4.7 Summary

In this chapter, we point out that, due to great variance of queries and different ranking characteristics of Web queries, single ranking model is not appropriate for diverse



(a) MQ2007 (b) MQ2008

Figure 14: Ranking performance (NDCG@K) of TRSVM against varying amount of noisy queries

types of queries. We employ a divide-and-conquer approach to learn multiple ranking functions according to diverse ranking characteristics of queries. We first identify ranking-sensitive query topics based on query features from pseudo feedbacks and prior knowledge of feature importance. Then, we propose a unified learning process to obtain ranking models corresponding to recognized query topics. An ensemble approach is applied to compute ranking for test queries by making use of multiple topic-specific ranking models. Experimental results illustrate that the proposed approach can significantly improve the ranking relevance over the single model approach and a straightforward local ranking approach, and the identified ranking-sensitive topics are more useful for improving ranking than pre-defined query categorization.

In Chapter 3 and 4, we explore the context information of queries from two specific aspects and investigate how to incorporate such context information into the task of learning to rank, respectively. The experimental results on both public benchmarks and commercial search engine dataset with deeper analysis have demonstrated that the context information of queries can be essential to improve the ranking relevance since it can provide much invaluable information for better understanding users' information need.

In the next part of the dissertation, we will take much deeper analysis on the context information of Web content, and explore whether this context information can benefit the search performance as well.

CHAPTER V

RANKING OVER SOCIAL MEDIA

From this chapter, we start investigating how to extract the context information of the Web content and take advantage of such context to improve the search performance. We will target at searching over the social media service and study how to explore the context of social media content to build effective search system over the social media service. In this chapter, we first propose a new general framework for searching social media content, which incorporates both the content features and the context of user interactions. Then, in chapter 6, a semi-supervised framework is proposed to explicitly compute content quality and user reputation in social media. These new context information will be incorporated into the general search framework to improve the search quality. Chapter 7 will discuss the spam of the context information and introduce a method to train more robust ranking models. Finally, Chapter 8 investigates techniques for integrating and organizing Web content in social media and studies how to use the extracted structured semantics as context to improve the performance of information retrieval.

5.1 Community Question Answering

Online social media content and associated services comprise one of the fastest growing segments on the Web. Such content includes social bookmarking (e.g. Delicious), social photo sharing (e.g. Flickr), social video sharing (e.g. Youtube), and many other forms of user-generated content. This content is different from the traditional content on the web (Web pages) in style, quality, authorship, and explicit support for social graphs. The explicit support for social interactions between users, such as posting comments, rating content, and responding to questions and comments makes

the social media unique and requires new techniques for analyzing and retrieving relevant content.

Question-Answering (henceforth QA) is a form of information retrieval where the users' information need is specified in the form of a natural language question, and the desired result is self-contained *answer* (not a list of documents). QA has been particularly amenable to social media, as it allows a potentially more effective alternative to web search by directly connecting users with the information needs to users willing to share the information directly. Some very successful examples of Community Question Answering (CQA) sites are Yahoo! Answers, and Naver[1]. In these portals users can express specific information needs by posting questions, and get direct responses authored by other web users, rather than browsing results of search engines. Both questions and responses are stored for future use by allowing searchers to first attempt to locate an answer to their question, if the same or similar question has been answered in the past. As CQA portals grow in size and popularity, searching for existing answers for a given question becomes increasingly crucial to avoid duplication, and save time and effort for the users. For example, Yahoo! Answers now has tens of millions of users, and stores hundreds of millions of answers to previously asked questions. These databases of questions and respective answers is an invaluable resource for specific information needs not well served by general-purpose search engines.

In particular, today's search engines are not yet generally capable to automatically generate brief but precise summaries that integrate information from multiple sources, or to answer queries that require deep semantic understanding of the query or the document. For example, consider a query "When is the hurricane season in the Caribbean?", which we submit to both Yahoo! Answers and Yahoo! Web search engine. Figure 15 and 16 show the best answer and top 5 search results respectively.

[1]http://www.naver.com/

Figure 15: The question "When is hurricane season in Caribbean?" in Yahoo! Answer and its best answer.

From those two figures, it is easy to see that Yahoo! Answers supply one brief answer but with high quality while for Yahoo! Search results people still need to click into the webpage to find needed information. Thus, CQA sites and search services provides an alternative channel for obtaining information more quickly and more precisely on the web.

However, finding relevant answers of a new query in CQA archives is a difficult task

Figure 16: Top 5 results when searching "When is hurricane season in Caribbean?" on Yahoo! Search.

that is distinct from web search and web question answering. The main differences are the availability of explicit user feedback and interactions, explicit authorship and attribution information, and organization of the content around topical categories and past question threads. Furthermore, the quality of the content varies even more than the quality of the traditional web content. A large fraction of the content reflects often unsubstantiated opinions of users, which are not useful for factual information retrieval and question answering. Hence, retrieving the correct factual answers to a question requires determining both relevance and quality of answer candidates.

As has been shown in recent work, user interactions, in particular explicit feedback from users, can provide a strong indication of the content quality [3]. Examples of such feedback include the selection of the "best" answer by the original question author, as well as the "thumbs up" and "thumbs down" ratings from other users. What is not clear from previous work is how to integrate explicit user feedback information and relevance into a unified ranking framework to retrieve answers that are relevant, factual, and of high quality.

This chapter presents a ranking framework to take advantage of user interaction information to retrieve high quality relevant content in social media. And, this study focuses on Community Question Answering to explore how to integrate relevance, user interaction and community feedback information to find the right factual, well-formed content to answer a user's query.

In summary, retrieving correct, factual, and well-formed answers from CQA archives is a crucial and challenging task. This work makes significant progress towards this goal, allowing the search system to find facts in the crowd with accuracy substantially higher than the current state-of-the-art.

5.2 Learning Ranking Functions for CQA Retrieval

Ranking functions are at the core of an effective CQA retrieval system. In this section, we will explore a learning-based approach to the design of the ranking functions. We will focus on the specific characteristics of Yahoo! Answers and discuss how to employ user interactions and community-based features in Yahoo! Answers to rank the answers. We start with a more precise definition of the problem of CQA retrieval, and then describe the different interactions available in a state-of-the-art CQA portal. Then we discuss how to represent textual elements and community-based elements in Yahoo! Answers as a vector of features. We then show how to extract preference data on answers from user interactions with Yahoo! Answers. Based on the extracted features and preference data, we apply the regression-based gradient boosting framework [82] to the problem of learning ranking function for CQA retrieval.

5.2.1 Problem Definition of QA Retrieval

In CQA systems, there are a very large amount of questions and answers posted by a diverse community of users. One posted question can solicitate several answers from a number of different users with varying degree of relevance to the posted question.

We abstract the social content in CQA system as a set of question-answer pairs:

$$\langle qst_i, ans_i^j \rangle$$

where qst_i is the $i-$th question in the whole archive of the CQA system and ans_i^j is the $j-$th answer to this question.

Given a user query, our goal is to order the set of question-answer pairs according to their relevance to the query, and the ordering is done by learning a ranking function for triples of the form,

$$\langle qr_k, qst_i, ans_i^j \rangle,$$

where qr_k is the k-query in a set of queries. We will first discuss how to effectively extract features and preference data for each triple of the above form and then discuss a regression-based gradient boosting methods for learning a ranking function for CQA retrieval.

5.2.2 User Interactions in Community QA

Community Question Answering (CQA) is a particularly popular form of social media, drawing millions of users to ask and answer each others' questions. Unlike other social media services, CQA services such as Yahoo! Answers provide distinct types of interactions among users that are specific to the CQA domain. Users in Yahoo! Answers do not only ask and answer questions, but also actively participate in regulating the system. A user can vote for answers of other users, mark interesting questions and even report abusive behavior. Therefore, a Yahoo! Answers user has a threefold role: asker, answerer and evaluator. And there are respectively three types of user interaction activities: asking, answering and evaluating. We summarize elements and interactions in Yahoo! Answers in Figure 17. The rest of this paper will focus on how to utilize this information for CQA retrieval.

In addition to facilitating the community question answering process, a CQA system must support effective search of the archives of past questions and answers. In

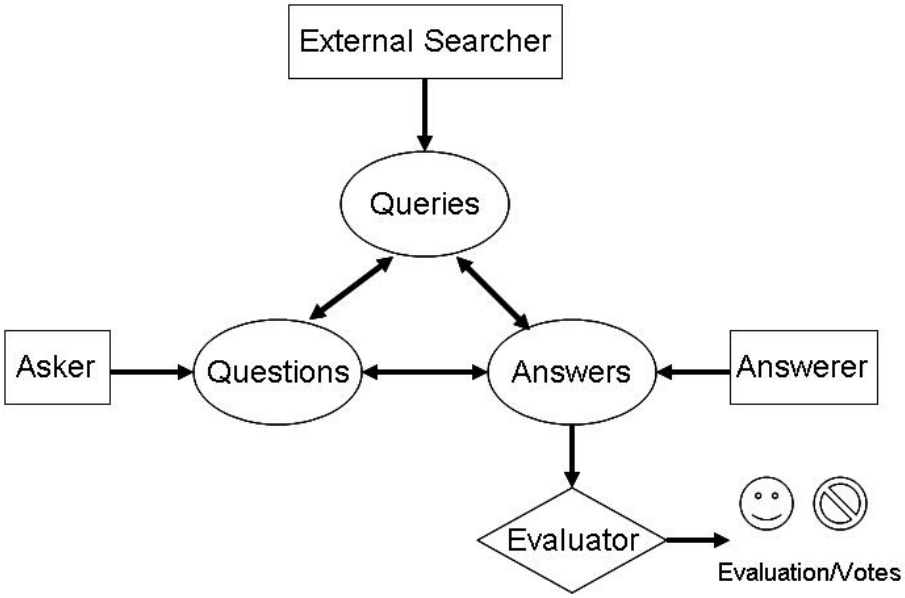


Figure 17: Elements and interactions in Yahoo! Answers: Askers post questions on Yahoo! Answers; Several users-answerers read questions and supply their answers to them; Users can also read these answers and give evaluations/votes; External users can submit queries to Yahoo! Answers and receive relevant questions with answers.

fact, one benefit of Yahoo! Answers to users and web searchers is precisely the immense archive of hundreds of millions of answers to past questions, with the associated community feedback. Searching this archive allows users the benefit of community feedback and precise and specific answers for many information needs not supported by general web search. However, because of the disparity in answer quality, and the difficulty in mapping user information needs to past questions, the problem of retrieving both *relevant* and *high quality* factual answers requires the integration of both content features (to estimate relevance) as well as user interaction and community features (to estimate quality). Having introduced our setting, we now describe our features for answer retrieval.

5.2.3 Features and Preference Data Extraction

We follow the general practice in information retrieval and represent each query-question-answer triple (qr, qst, ans) as a combination of textual features (i.e., textual

similarity between query, question and answers), statistical features (i.e., independent features for query, question and answers) and social features (i.e., user interaction activities and community-based elements). In Yahoo! Answers, there is an additional important type of user feedback — user evaluation in the form of votes (represented as the "thumbs up" and "thumbs down" metaphors). We can use this information to infer preference relevance judgments for the set of answers. In the following, we discuss features and preference data extraction in more details.

Table 10: Features used to represent textual elements and user interactions

Textual Features of Questions, Answers and Queries	
Query-Question Overlap	Overlapping terms between query and question
Query-Answer Overlap	Overlapping terms between query and answer
Question-Answer Overlap	Overlapping terms between question and answer
Query-Question length ratio	Ratio between number of tokens in query and question
Query-Answer length ratio	Ratio between number of tokens in query and answer
Question-Answer length ratio	Ratio between number of tokens in question and answer
Statistical Features of Questions, Answers and Queries	
Query Length	Number of tokens in query
Question Length	Number of tokens in question
Answer Length	Number of tokens in answer
Question Lifetime	How long has this question been posted
Answer Lifetime	How long has this answer been posted
Yahoo! Question Rank	The rank of question in Yahoo! Answers
Yahoo! Answer Rank	The rank of answer in Yahoo! Answers
Question Popularity	How many answers are received under the question
Votes Number	Number of votes for answer
User Interaction/Social Elements Features	
Asker Total Points	Points calculated by Yahoo! based on Asker's activity history
Asker Total Answers	How many answers did the asker submit?
Asker Best Answer	How many best answers did the asker propose?
Num of Questions Asked by Asker	How many questions did the asker post?
Num of Questions Resolved by Asker	How many questions were resolved by the asker?
Asker stars received	How many stars does the asker receive in Yahoo! Answers
Answerer Total Points	Points calculated by Yahoo! based on Answerer's activity history
Answerer Total Answers	How many answers did the Answerer submit?
Answerer Best Answer	How many best answers did the Answerer propose?
Num of Questions Asked by Answerer	How many questions did the Answerer post?
Num of Questions Resolved by Answerer	How many questions were resolved by the Answerer?
Answerer stars received	How many stars does the Answerer receive?

5.2.3.1 Representing Textual Elements and User Interactions as Features

Textual Elements Features: As we mentioned before, there are three textual elements in Yahoo! Answers: questions, answers and queries (showed in Figure 17). We first design features from each of these three elements independently, such as "number of tokens for a query" for query, "how long has the question been posted"

for question, "number of received votes for an answer" for answer etc. (We describe them as statistical features in Table 10)

Then, we also extract textual features from relationship between questions, answers and queries. For example, the number of overlapping terms and token number ratio between two of these three elements. We also introduce other features describing similarity between these three elements. (We describe them as textual features in Table 10)

User Interaction Features: As discussed before, there are three kinds of roles each user in CQA system may play, namely Asker, Answerer and Evaluator. Figure 17 shows the interactions between these three roles. Askers post their questions on CQA system in the hope that other users answer these questions. Answerers submit their answers to the question in the QA system. Some answers have high quality while the majority are not useful. Evaluators give positive and negative votes for an answer after they read an answer and related question in the CQA system. In the community of CQA system, each user can play all of these three roles at the same time.

For each user in the user community of a CQA system, there are several features to describe his or her activities, such as the number of questions he or she asked, the number of answers he or she posted, the number of best answers he or she posted etc. These features to certain extent can approximate the user's expertise in the QA community. And user's expertise within certain topics can in turn indicate the quality of his or her answers to the questions about certain topics. For example, in a query-question-answer triple, if answerer tend to post useful answers or even best answers in the past, he or she is more likely to give answers of high quality this time. Similarly reputation of askers and evaluators can also indicate quality of answers. Therefore, in each query-question-answer triple, we also extract features indicating user's activities in the CQA system. As there is no information about evaluators in Yahoo! Answer, we only consider features for askers and answerers, such as "number

of questions the asker asked in the community", "number of best answers the answerer posted in community". These features are listed in Table 10.

5.2.3.2 Representing Users Evaluations as Preference Data

One of the user interactions in a CQA community, especially in Yahoo! Answers, is their evaluations for the existing answers. In Yahoo! Answers, after reading existing answers for a question, user can give his or her judgment as the evaluation for the answers. If he or she considers the answer as useful and of high quality, he or she can add a plus vote to this answer. Otherwise, a minus votes may be added to the answer.

We examine users evaluation data and extracted a set of preference data which can be used for ranking the answers as follows. For each query qr, under the same question qst, we consider two existing answers ans_1 and ans_2 from Yahoo! Answers. Assume that in the users evaluation data, ans_1 has p_1 plus votes and m_1 minus votes out of n_1 impressions while ans_2 has p_2 plus votes and m_2 minus votes out of n_2 impression. We want to consider answer pairs ans_1 and ans_2 to see whether ans_1 is preferred over ans_2 in terms of their relevance to the question qst. To this end, we assume that plus votes obey binomial distribution, showed as following:

$$B(k; n, p) = \begin{pmatrix} n \\ k \end{pmatrix} p^k (1-p)^{n-k}$$

We use the approach in [82] and apply likelihood ratio test to examine whether a pair of answers is significant or not, i.e., whether there are enough votes to compare the pair. In particular we compute the following statistic,

$$\lambda = \frac{B(p_1 + p_2; n_1 + n_2, (p_1 + p_2)/(n_1 + n_2))}{B(p_1; n_1, p_1/n_1) B(p_2; n_2, p_2/n_2)} \rightarrow -\chi^2$$

For a pair of answers ans_1 and ans_2, when the above value is greater than a threshold, we say the pair is significant. If ans_1 and ans_2 form a significant pair, we then extract

a preference for the pair by comparing $\frac{p_1}{p_1+m_1+s}$ with $\frac{p_2}{p_2+m_2+s}$, where s is positive constant, i.e., if the former value is bigger than the later one, then we say ans_1 is preferred over ans_2 which is denoted by $ans_1 \succ ans_2$, and vice versa.

5.2.3.3 Representing Labeled data as Preference Data

Besides users evaluation information, we can also extract preference data from labeled data. For two query-question-answer items with the same query,

$$(qr, qst_1, ans_1) \quad (qr, qst_2, ans_2),$$

let their feature vectors be x and y, respectively. If ans_1 has a higher labeled grade than ans_2, we include the preference $x \succ y$ while if ans_2 has a higher labeled grade than ans_1, we include the preference $y \succ x$. We will discuss how to obtain labeled data in details in section 5.3.1.

5.2.4 Learning Ranking Function from Preference Data

Once the features and preference data are extracted, the next question is how to use them for the purpose of learning a ranking function for CQA retrieval. We apply a framework for solving ranking problems from preference data developed in [82]. This framework proposes a squared hinge loss function for learning ranking functions from preference data; it also presents an algorithm that adapts functional gradient descent for minimizing the loss function. We now briefly describe the basic idea of the algorithm in [82].

Suppose the set of available preferences is

$$\mathcal{S} = \{\langle x_i, y_i \rangle \mid x_i \succ y_i, i = 1, ..., N\}.$$

Here each $\langle x, y \rangle \in S$, x, y denote the feature vectors for two query-question-answer triples with the same query. $x \succ y$ means that x is preferred over y, i.e. x should be ranked higher than y. In other words, the answer in x is considered more relevant than that in y with respect to the same query in both triples.

In [82], the problem of learning ranking functions is cast as the problem of computing a function h, such that h match the given set of preferences, i.e., $h(x_i) \geq h(y_i)$, if $x_i \succ y_i$, $i = 1, ..., N$, as much as possible. The following objective function (squared hinge loss function) is used to measure the risk of a given ranking function h,[2]

$$\mathcal{R}(h) = \frac{1}{2} \sum_{i=1}^{N} (\max \{0, h(y_i) - h(x_i) + \tau\})^2,$$

and we need to solve the following minimization problem

$$\min_{h \in \mathcal{H}} \mathcal{R}(h),$$

where $\mathcal{H}$ is a function class, chosen to be linear combinations of regression trees in our case. The above minimization problem is solved by using functional gradient descent discussed in [23]. We summarize the algorithm for learning ranking function h using gradient boosting (GBrank) in Algorithm 2.

Two parameters need to be determined: the shrinkage factor and the number of iterations, this is usually done by cross-validation [82].

5.3 Experimental Setup

This section presents our evaluation setup. First we describe our dataset including the queries and the corresponding corpus of questions and answers. Then we describe our evaluation metrics (Section 5.3.2) and the ranking methods to compare (Section 5.3.3) for the experimental results reported in Section 5.4.

5.3.1 Datasets

Factoid questions from the TREC QA benchmarks

We use factoid questions from seven years of the TREC QA track evaluations (years

[2]This loss function can be considered as a smooth surrogate of the total number of contradicting pairs in the given preference data with respect to the function h. We say $x \succ y$ is a contradicting pair with respect to h if $h(x) < h(y)$.

Algorithm 2: GBrank

input : traing dataset: $\mathcal{S} = \{\langle x_i, y_i \rangle\}$
output: the ranking function h
Start with an initial guess h_0, for $k = 1, 2, ...$
begin
  while *h not converge* **do**
    •Using h_{k-1} as the current approximation of h, we separate $\mathcal{S}$ into two disjoint sets,

$$\mathcal{S}^+ = \{\langle x_i, y_i \rangle \in \mathcal{S} | h_{k-1}(x_i) \geq h_{k-1}(y_i) + \tau\}$$

    and

$$\mathcal{S}^- = \{\langle x_i, y_i \rangle \in \mathcal{S} | h_{k-1}(x_i) < h_{k-1}(y_i) + \tau\}$$

    •Fit a regression function $g_k(x)$ using Gradient Boosting Tree [23] and the following training data

$$\{(x_i, h_{k-1}(y_i) + \tau), (y_i, h_{k-1}(x_i) - \tau) | \langle x_i, y_i \rangle \in \mathcal{S}^-\}$$

    •Form the new ranking function as

$$h_k(x) = \frac{k h_{k-1}(x) + \eta g_k(x)}{k+1}$$

    where η is a shrinkage factor.
end

1999–2006)[3] for the experiments reported in Section 6. It is worth noting that TREC questions from the years 1999 to 2003 are independent of each other: each question is self-contained and we submit directly as the query. Starting from 2004, however, the questions are organized in groups with a 'target'. For those questions, we submit their 'target' as well as the questions themselves. In total, approximately 3,000 factoid TREC questions were compiled as the initial set of queries.

Since we need *some* candidate answers from Yahoo! Answers to estimate how well different ranking functions perform, we select the 1250 TREC factoid questions that have at least one similar question in the Yahoo! Answers archive.

Question-answer collection dataset

[3]http://trec.nist.gov/data/qa.html

Our dataset was collected in order to simulate a user's experience with a community QA site. We submit each TREC query to the Yahoo! Answers web service[4] and retrieve up to 10 top-ranked related questions according to the Yahoo! Answers ranking. For each of these Yahoo! questions, we retrieve as many answers as there are available for each question thread. There are, in total, 89642 $\langle query, question, answer \rangle$ tuples. 17711 tuples (19.8%) are labeled as "relevant" while 71931 (81.2%) are labeled as non-relevant.

Relevance Judgments

In our experiment, the data are labeled in two ways: by using the TREC factoid answer patterns, and, independently, manually in order to validate the pattern-based automatic labels.

For automatic relevance labels we use the available regular expression answer patterns for the TREC factoid questions. We check every answer's text body, and if the text matches one of the answer patterns, we consider the answer text to be relevant, and non-relevant otherwise.

In order to validate the accuracy of our automatically-assigned relevance labels, we independently labeled a number of answers by hand. The manually labeled answers were compared with the automatically generated labels, resulting in over 90% agreement between the automatic and manual methods. In the cases of disagreements were due to the excessive strictness of the answer patterns, and to the world changing (e.g., with a different correct answer for a question "Who is the prime minister of Japan."). This is not surprising, as some of the answer patterns were created years ago around the time of the TREC QA evaluation. In summary, automatically generated labels, even though with some small degree of noise, nevertheless exhibit high agreement with manual relevance judgments, and serve as a good proxy for comparing rankings.

[4] http://developer.yahoo.com/answers/

5.3.2 Evaluation Metrics

We adapt the following information retrieval metrics to evaluate the performance of the ranking function.

- **Mean Reciprocal Rank(MRR)**: The MRR of each individual query is the reciprocal of the rank at which the first relevant answer was returned, or 0 if none of the top N results contained a relevant answer.The score for a sequence of queries is the mean of the individual query's reciprocal ranks. Thus, MRR is calculated as

$$MRR = \frac{1}{|Qr|} \sum_{q \in Qr} \frac{1}{r_q}$$

  where Qr is a set of test queries, r_q is the rank of the first relevant document for q.

- **Precision at K**: for a given query, $P(K)$ reports the fraction of answers ranked in the top K results that are labeled as relevant. In our setting, we require a relevant answer to be labeled "matched" for TREC pattern. For this metric, the position of relevant answers within the top K is irrelevant, while it measures overall user potential satisfaction with the top K results.

- **Mean Average of Precision(MAP)**: Average precision for each query is defined as the mean of the precision at K values calculated after each relevant answers was retrieved. The final MAP value is defined as the mean of average precisions of all queries in the test set. This metrics is the most commonly used single-value summary of a run over a set of queries. Thus, MAP is calculated as

$$MAP = \frac{1}{|Qr|} \sum_{q \in Qr} \frac{\sum_{r=1}^{N} (P(r) \times rel(r))}{|R_q|}$$

  where Qr is a set of test queries, R_q is the set of relevant document for q, r is the rank, N is the number retrieved, $rel()$ is a binary function on the relevance

of a given rank, and $P()$ is precision at a given cut-off rank.

5.3.3 Ranking Methods Compared

To evaluate the Q&A retrieval quality, we compare the quality of following methods:

- Baseline_Yahoo(baseline1): In this method, we adapt default ranking in Yahoo! Answers. Answers to a particular question are ordered by posting date. The older one is ranked higher except that best answers always come first.

- Baseline_Votes(baseline2): In this method, similar to our baseline1, we let best answers always be on top of the answer list. However, following answers are ranked in decreasing order by number of (positive votes - negative votes) received. If there is no vote for some answers, we order them by Yahoo! default ranking.

- GBRank: Ranking function with community/social features: this is our method presented in Section 5.2.4

For Yahoo! Answers, since we first get a list of Yahoo! questions for one TREC query, and each of these Yahoo! questions has its own answers, there are multiple alternatives for calculating MRR, Precision and MAP values for *Baseline_Yahoo* and *Baseline_Votes*. First, we need to introduce some nomenclature: for each TREC query TQ, we retrieve a list of related questions from Yahoo! Answers $YQ_a, YQ_b...$ (as shown in Figure 18). After clicking on one of these questions, we get the answers to each question, e.g., YQ_a, as $YQ_a^1, YQ_a^2...YQ_a^n$, as shown in Figure 15:

- **MRR_MAX**: Calculate MRR value for each YQ_a,

  $YQ_b...$ and use the highest value as this TQ's MRR. This baseline simulates an "intelligent" user who always selects the most relevant retrieved Yahoo! question thread first (as measured by the corresponding MRR for the thread).

- **MRR_STRICT**: Calculate MRR value for each YQ_a, YQ_b... and use the their average value as this TQ's MRR. This baseline simulates a user who blindly follows the Yahoo! Answer's ranking and examines retrieved question threads and corresponding answers in the order they were originally ranked.

- **MRR_RR**(round robin): Use YQ_a's first answer as TQ's first answer, YQ_b' first answer as TQ's second answer and so on, then calculate this TQ's MRR. This baseline simulates a "jumpy" user who believes that answers that come first, no matter to which questions, are always better, and jumps between question threads looking at the top-ranked answers for each tread in order of the original ranking.

Figure 18: Top 5 results when searching "When is hurricane season in Caribbean?" on Yahoo! Answers.

The variants for the other two metrics, Precision and MAP (namely, **PREC_MAX, PREC_STRICT, PREC_RR, MAP_MAX, MAP_STRICT, MAP_RR**), are calculated similarly.

In summary, MRR_MAX (and PREC_MAX and MAP_MAX) represent the upper bound on Yahoo! Answers' accuracy (with the current retrieval and ranking algorithms) from the perspective of an intelligent searcher. This is an extremely strong family of baseline metrics, as it assumes an "oracle" that always makes the right decision to click on the question threads that contain the correct answer in the highest ranking position.

5.4 Experimental Results

5.4.1 Learning Ranking Function

To learn the ranking function, we generate the training and testing data as follows: we randomly select 400 TREC queries from total 1250 TREC queries and collect all the related QA for these 400 queries. We use ten-fold cross validation to perform the training of the proposed ranking function using the algorithm introduced above. Ten-fold cross validation involves dividing the judged data set randomly into 10 equally-sized partitions, and performing 10 training/testing steps, where each step uses 9 partitions to train the ranking function and the remaining partition to test its effectiveness. Note that the following results were done on this smaller set train the ranking function - the main evaluation will be performed in the next section, over the remaining 850 TREC questions that were not used in training in any way.

Figure 19 reports the Precision at K for the hold-out validation data against each iteration of our learning algorithm. It can be clearly seen that Precision increases for the first 60 iterations, after which the algorithm converges and additional iterations are not helpful.

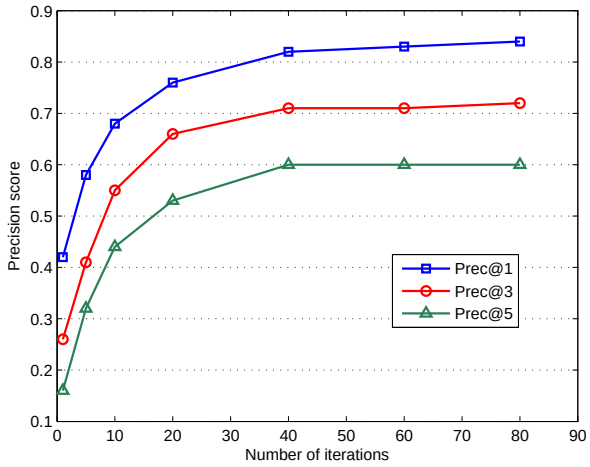


Figure 19: Precision at 1, 3, 5 for testing queries against GBrank iterations

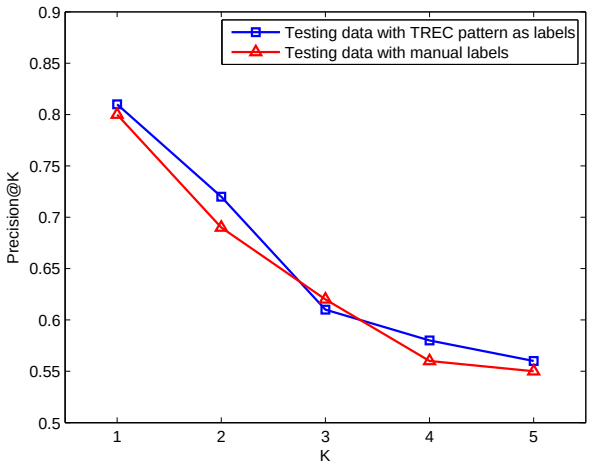


Figure 20: Precision at K for testing queries with manual labels and labels generated by TREC pattern

5.4.2 Robustness to Noisy Labels

As mentioned in Section 5.3.1, the relevance judgments was obtained by matching an answer with TREC answer patterns. We have also found that 90% of items were given the same labels under both manual labeling and TREC pattern labeling and the remaining 10% of automatic labels were erroneous. To show that our learning framework is robust, we experiment with training on the noisy automatically generated labels, and testing on the smaller set of "gold standard" manually assigned relevance

labels. For this experiment we used 50 queries (testing data) which have been labeled manually. Then, we randomly select the other 350 TREC queries (training data) and related questions and answers to train the ranking function. Figure 20 shows the Precision at K for testing data based on manual labels and TREC pattern labels respectively. While the accuracy when evaluating against manual labels is slightly lower than automatic labels, the differences are not substantial, which implies that our algorithm still generates a nearly-optimal model even when trained on noisy relevance labels. Furthermore, the high correspondence of automatic and manual label accuracy results validates our decision to use only automatic labels for the remaining experiments, to enable experiments on the larger scale.

5.4.3 Ablation Study on Feature Set

To gain a better understanding of the important features for this domain we perform an ablation study on our feature set to explore which features are significant to answers ranking.

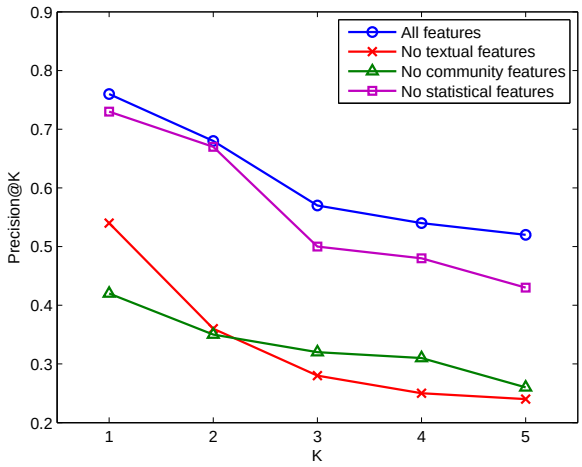


Figure 21: Precision at K for feature ablation study

As shown in Table 10, there are three major categories of our feature set: textual features, community features and statistical features. Figure 21 reports the Precision at K when learning ranking function with removing each category respectively. It

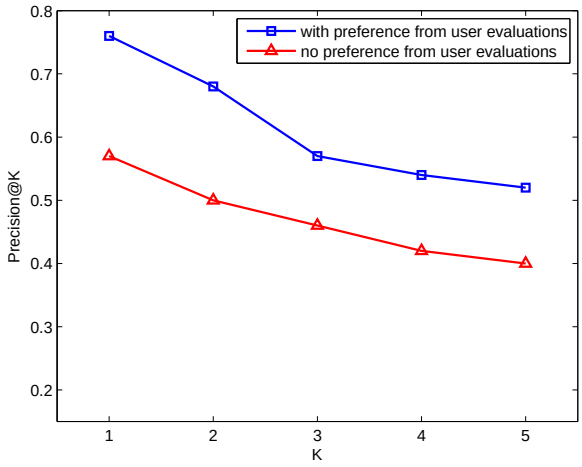


Figure 22: Precision at K without incorporating user evaluations as preference data

is easy to see that removing both textual features and community features cause a significant decreasing on precision. While there is a slight reduction on precision when removing statistical features, it is clear that these features are not as important as the textual and community features. Interestingly, textual features are less important for Precision at 1. We hypothesize that for the top result it is more important for an answer to be chosen as "best" by the asker (one of the community features), than to have appropriate textual characteristics.

In addition, we also test the effectiveness of preference data from users evaluations. In order to test its effectiveness, we learn a ranking function without incorporating preference data from users evaluations. Figure 22 shows the Precision at K of this new ranking function. From this figure, we can see that users evaluations play a very important role in learning ranking function.

5.4.4 QA Retrieval

In this experiment, we train our ranking function on the whole training data (i.e., the 400 TREC queries from the previous experiments) and test on the remainder hold-out data of 850 TREC queries and associated community QA pairs.

Figures 23 and 24 illustrate the Precision at K of GBrank compared with the

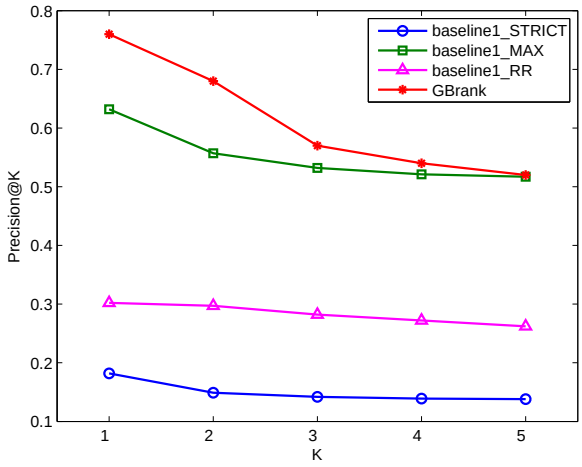


Figure 23: Precision at K for GBrank, baseline1_MAX, baseline1_RR and baseline1_STRICT for vary K

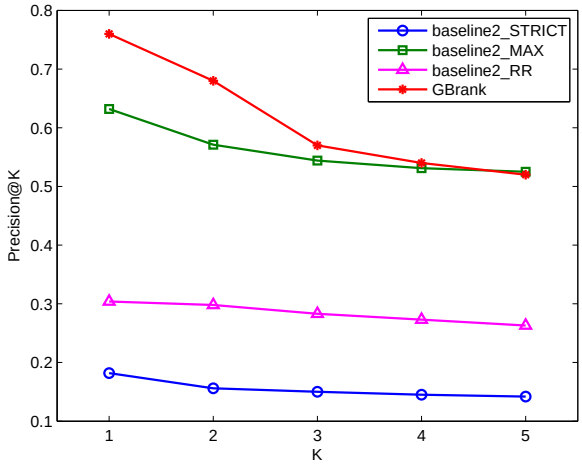


Figure 24: Precision at K for GBrank, baseline2_MAX, baseline2_RR and baseline2_STRICT for vary K

two baseline methods. These figures show that the precision of two baseline methods are nearly the same, while GBrank out-perform both of them. In particular, the Precision at 1 of GBrank is 76%, compared to 63% precision at 1 exhibited by the _MAX baselines. We can also see that PREC_MAX performs better than PREC_RR and PREC_STRICT, and GBrank has the similar performance with PREC_MAX the values of K larger than 3.

In Table 11 and 12, we illustrate the MAP and MRR scores for two baseline methods as well as GBrank. From these two tables, it is clear that MAX can perform better than the other two metrics for baseline, but GBrank reaches much better performance than all the metrics for two baseline methods. In particular, GBrank achieves a gain of about 18% relative to the _MAX metrics.

Table 11: Mean Reciprocal Rank (MRR) and Mean Average Precision (MAP) for GBrank and other metrics for baseline1

	MRR	Gain	MAP	Gain
STRICT	0.213	0.569	0.043	0.422
ROUND ROBIN	0.401	0.381	0.145	0.310
MAX	0.662	0.120	0.441	0.024
GBrank	0.782	-	0.465	-

Table 12: Mean Reciprocal Rank (MRR) and Mean Average Precision (MAP) for GBrank and other metrics for baseline2

	MRR	Gain	MAP	Gain
STRICT	0.214	0.568	0.045	0.420
ROUND ROBIN	0.403	0.379	0.159	0.306
MAX	0.664	0.118	0.443	0.022
GBrank	0.782	-	0.465	-

To understand how GBRank can outperform an "oracle" baseline, consider that the ordering of answers within a question thread remains fixed (either by date – as the default, or by decreasing votes). In contrast, GBrank obtains a better ranking of answers within each question thread, as well as a global ranking of all answers. Then, improved ranking within each Yahoo questions thread contributes to the higher score than MRR_MAX. Overall, applied on Yahoo! Answers, our proposed framework achieves a significant improvement on the performance of QA retrieval over the Yahoo! Answers' default ranking and the supported optional votes-based ranking. In addition, from the experiment, we can find that our method is able to retrieve relevant answers at the top of results. In summary, we have shown that GBRank significantly outperforms extremely strong baselines, achieving precision at 1 of over 76% and

MRR of over 0.78, which are high values even for traditional factoid QA.

5.5 Summary

Community Question Answering (CQA) is transforming the way people search for information. We have presented a robust, effective method for retrieving factual answers from CQA archives, and have demonstrated our method to be significantly more effective than the best possible accuracy a user can achieve when interacting with the current state-of-the-art question search on a major CQA site. Furthermore, our large scale experiments demonstrate that our method is robust to noise in the automatically generated training preference judgments. We have complemented our study with an analysis of the results to gain insight into the significant dimensions of fact retrieval from social media. In particular, we found that textual features and community features are crucial, and that user feedback, while noisy, provides sufficient relevance judgment to improve the learning of the ranking functions.

By significantly improving the accuracy of retrieving well-formed, factual answers, our work has the potential to transform how users interact with CQA sites; to improve the experience by reducing duplicate questions; and to better integrate question answering and search over CQA archive with the mainstream web search results. In summary, our work is a crucial component for factual information seeking in the increasingly important social media environment.

Although we have used community interaction features to estimate the content quality and user reputation, this chapter does not take deeper study on identifying these two kinds of context information explicitly. In the next chapter, we will propose to recognize reliable content and user and take advantage of such explicit context information to improve search performance.

CHAPTER VI

LEARNING TO RECOGNIZE RELIABLE USERS AND CONTENT IN SOCIAL MEDIA

6.1 Content Quality and User Reputation in Social Media

As discussed in the last chapter, social media service, such as CQA portals, have grown in size and popularity. For example, the popular CQA site, Yahoo! Answers, already has tens of millions of users, and stores hundreds of millions of answers to previously asked questions, and serves millions of visits each day. These databases of past questions and respective answers are proving to be a valuable resource for specific information needs not well served by general-purpose Web search engines. Unfortunately, the quality, accuracy, and comprehensiveness of the content in the CQA archives varies drastically, and a large portion of the content is not useful for answering user queries. Not surprisingly, the reputation and expertise of the contributors can provide crucial indicators into the quality and the reliability of the content. The reputation of the contributor could also be a valuable factor for ranking search results from CQA repositories, as well as for improving the system interface and incentive mechanisms.

In a CQA environment, schematically shown in Figure 25, there are three sets of connected entities: users, answers and questions. In addition to the intuitive connection between questions and answers, users are also connected with two other sets of entities by both expressing specific information needs via posting questions, and by responding to existing question via posting their answers to questions.

Unfortunately, existing methods for estimating content quality in CQA either require large amounts of supervision (e.g., [3]) or focus on the network properties

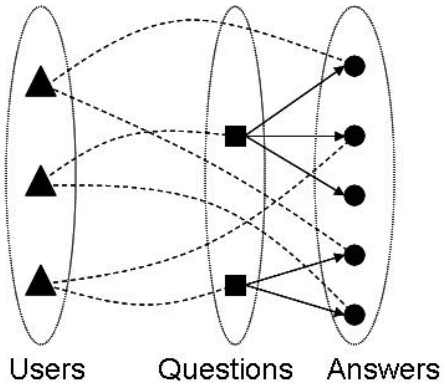


Figure 25: Network of interactions in CQA connecting users, questions and answers

of the CQA without considering the actual content of the information exchanged (e.g., [81]). We observe that user reputation and the quality of the content they produce are often tied together in a mutually reinforcing relationship. Building on this observation, this study proposes a general framework for simultaneously calculating the answer and question quality and user reputation based on their network relationships, coupled with the individual quality/reputation-related features. In the proposed framework, the CQA interactions are viewed as composite bipartite graphs where each pair of entity types (e.g., users and the answers they generate) can form one bipartite graph. In this study, we develop a machine learning approach that starts with a set of known labels for users and answers, and exploits the mutual reinforcement between the connected entities in each bipartite graph to compute the respective quality and reputation scores simultaneously, iteratively refining the labels for the unlabeled entities.

6.2 Learning Content Quality and User Reputation in CQA

Answer and question quality are crucial to information retrieval in community question answering. It has been noted in previous work that user reputation is expected to correlate with answer and question quality but the relationship between user reputation and content quality is not straightforward. "Authoritative" users may provide

poor answers, and "poor" users may occasionally provide excellent answers [3, 1]. In this section, we will explore a learning-based approach to calculate answer and question quality as well as user reputation in CQA, simultaneously. We will focus on the specific characteristics of Yahoo! Answers and discuss how to employ coupled mutual reinforcement principle to learn answer and question quality and user reputation. We will start with a more precise definition of the problem of calculating answer and question quality and user reputation, and then describe the mutual reinforcement principle between these three types of entities in CQA. Then we present a coupled mutual reinforcement framework to model answer and question quality and user reputation. Based on mutual reinforcement in CQA network, we apply a semi-supervised regression-based approach to the problem of learning answer and question quality and user reputation.

6.2.1 Problem Statement

In a CQA system, there are three distinct types of entities: users $\mathcal{U}$, answers $\mathcal{A}$, and questions $\mathcal{Q}$. Questions and answers are posted by a diverse community of users. And one question can solicit several answers from a number of different users. We can further categorize users into two subsets: askers $\mathcal{U}_q$ and answerers $\mathcal{U}_a$. Note that there can be an overlap between askers and answerers - that is, a user may post both questions and answers. Before proceeding, we define question and answer quality and user reputation more precisely:

Definition 1 Question Quality*: a score between 0 and 1 indicating a question's effectiveness at attracting high-quality answers.*

Definition 2 Answer Quality*: a score between 0 and 1 indicating the responsiveness, accuracy, and comprehensiveness of the answer to a question.*

In previous work, question and answer quality were defined in terms of content, form, and style, as manually labeled by paid editors [3]. In contrast, our definitions focus

on question effectiveness, and the answer accuracy – both quantities that can be measured automatically and do not necessarily require human judgments.

Definition 3 Answer-reputation*: a score between 0 and 1, indicating the expected quality of the* answers *posted by a user.*

Definition 4 Question-reputation*: a score between 0 and 1, indicating the expected quality of the* questions *posted by a user.*

Clearly, the definitions above are somewhat "circular" in that the reputation of the user depends on the quality of the questions or answers they post–where quality, in turn, can be influenced by the user reputation. In fact, we will soon show how to exploit this relationship in our mutual reinforcement framework. We now state our problem more formally:

Problem: Predicting Content and Contributor Quality

> Given a CQA archive, determine the quality of each question and answer and the answer-reputation and question-reputation of each user, simultaneously, with minimal manual labeling.

In the rest of this section we will first introduce the "coupled mutual reinforcement principle" for content quality and user reputation in community question answering. We will then present our novel semi-supervised, regression-based approach, based on the mutual reinforcement idea.

6.2.2 Coupled Mutual Reinforcement Principle

Recall that our goal is to identify high-quality questions and answers, and high-reputation users, simultaneously. We now state the *mutual reinforcement principle* that underlies our approach to solving this problem with the minimum of manual labeling:

An answer is likely to be of high quality *if the content is responsive and well-formed, the question has high quality, and the answerer is of high answer-reputation. At the same time, a user will have high* answer-reputation *if she posts high-quality answers, and high* question-reputation *if she tends to post high-quality questions. Finally, a question is likely to be of high quality if it is well stated, is posted by a user with high question reputation, and attracts high-quality answers.*

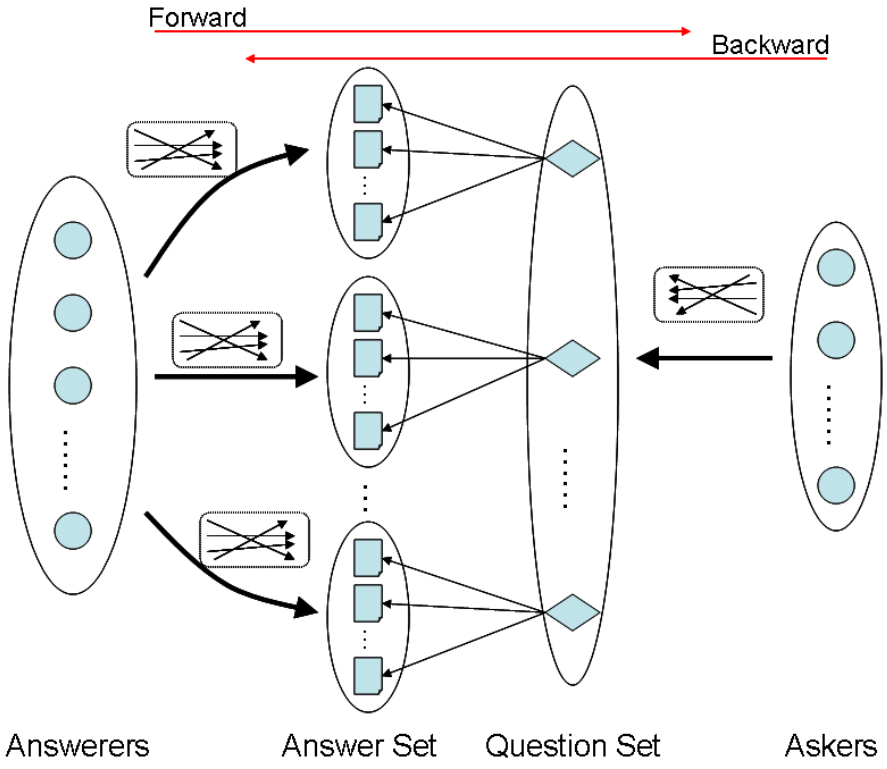


Figure 26: $|\mathcal{Q}|$ *coupled* bipartite graphs connecting with user-question bipartite graph.

Before we can turn this idea into an algorithm, we need to represent our setting more precisely. Recall that CQA systems are centered around three entities and their relationships: Users ($\mathcal{U}$), questions ($\mathcal{Q}$), and answers ($\mathcal{A}$). The relationships between these entities are illustrated in Figure 26. In particular, to represent the relationships between answers and their authors, we can use a bipartite graph with an edge between

each user and the answers that they post. Note that it is convenient to partition these graphs according to the *question thread* – that is, to consider subgraphs that involve answers to a particular question. Similarly, we can represent the relationship between askers and the questions they post by a bipartite graph with an edge connecting an asker to their posted question(s). We consider the sets of bipartite graphs that share the same question to be *coupled*.

We denote the whole graph in Figure 26 as

$$G(\mathcal{U}, \mathcal{A}, \mathcal{Q}, M_{\mathcal{UA}}, M_{\mathcal{UQ}}, M_{\mathcal{QA}})$$

where $M_{\mathcal{UA}} = [m_{ij}]$ is the $|\mathcal{U}|$-by-$|\mathcal{A}|$ matrix containing all the pairwise edges, i.e., $m_{ij} = 1$ if there is an edge between user $\mathbf{u}_i$ and answer $\mathbf{a}_j$. Similarly, $M_{\mathcal{UQ}}$ and $M_{\mathcal{QA}}$ are the matrices containing pairwise edges representing the association between users and questions they post, and question and the posted answers, respectively. Note that users may appear in both the asker and the answerer sets; however, we purposefully remove this additional coupling by separating the "asker" from the "answerer" personas of each user and modeling the reputation of each persona separately.

Now we can state the mutual reinforcement principle introduced earlier more precisely, as the set of four simultaneous equations governing the answer-reputation y_u^a and question-reputation y_u^q of a user u, and the corresponding answer quality y_a and question quality y_q respectively:

$$y_u^a \propto \sum_{u \sim a} m_{ua} y_a \tag{21}$$

$$y_a \propto \alpha \sum_{a \sim u} m_{ua} y_u^a + (1-\alpha) y_{q(\sim a)} \tag{22}$$

$$y_u^q \propto \sum_{u \sim q} m_{uq} y_q \tag{23}$$

$$y_q \propto \gamma \sum_{q \sim a} m_{aq} y_a + (1-\gamma) y_{u(\sim q)}^q \tag{24}$$

where $u \sim a$ or $u \sim q$ represents an edge between a user and her answers, or user and her questions; $y_{q(\sim a)}$ denotes the quality of answer a's question; $y_{u(\sim q)}^q$ denotes

the question-reputation of the user who ask question q; the symbol $\propto$ stands for "proportional to"; and α and γ are proportionality constants.

To simplify the notation, we collect answer-reputation and question-reputation scores of all users into vectors $\mathbf{y}_u^a$ and $\mathbf{y}_u^q$ respectively, and collect all answer and question quality scores into vector $\mathbf{y}_a$ and $\mathbf{y}_q$, resulting in the simplified form of the same principle:

$$\mathbf{y}_u^a = M'_{\mathcal{UA}}\mathbf{y}_a \tag{25}$$

$$\mathbf{y}_a = \alpha M_{\mathcal{UA}}^T\mathbf{y}_u^a + (1-\alpha)M_{\mathcal{QA}}^T\mathbf{y}_q \tag{26}$$

$$\mathbf{y}_u^q = M'_{\mathcal{UQ}}\mathbf{y}_q \tag{27}$$

$$\mathbf{y}_q = \gamma M_{\mathcal{UQ}}^T\mathbf{y}_u^q + (1-\gamma)M_{\mathcal{QA}}\mathbf{y}_a \tag{28}$$

where M^T stands for the matrix transpose of M; $M'_{\mathcal{UA}}$ and $M'_{\mathcal{UQ}}$ is derived from $M_{\mathcal{UA}}$ and $M_{\mathcal{UQ}}$ as for each $m'_{ij} \in M'_{\mathcal{UA}}$, $m'_{ij} = \frac{m_{ij}}{\sum_{j=1}^{|\mathcal{A}|} m_{ij}}(m_{ij} \in M_{\mathcal{UA}})$ and for each $m'_{ij} \in M'_{\mathcal{UQ}}$, $m'_{ij} = \frac{m_{ij}}{\sum_{j=1}^{|\mathcal{Q}|} m_{ij}}(m_{ij} \in M_{\mathcal{UQ}})$.

We can now turn the mutual reinforcement principle into a semi-supervised algorithm to estimates content quality and user reputation, as we describe next.

6.2.3 CQA-MR: Coupled Semi-Supervised Mutual Reinforcement

Due to the tight correlation and connection between those three sets of entities in CQA (questions, answers and users), we apply a mutually reinforcing approach to learn the question-reputation and answer-reputation of users as well as the quality of questions and answers, simultaneously. In the following, we first describe the features for learning question and answer quality and user reputation (Section 6.2.3.1). Then, we present a logistic regression approach for learning question and answer quality and user reputation (Section 6.2.3.2). However, we are given very few labels on answers and questions quality and users reputation in CQA. Thus, we apply the discussed coupled mutual reinforcement relationship for semi-supervised learning on answers and questions quality and users reputation, and such relationship is also incorporated

into the log-likelihood function (Section 6.2.3.2). Finally, we summarize a **CQA-MR** algorithm which can both fit the model and learn on answer and question quality and users reputation (Section 6.2.3.3).

6.2.3.1 Features

In a CQA system, there are several complementary feature sets for answers, questions and users, respectively. Table 13 shows a list of features for answers, questions and users, which form the feature space of answers, $X(\mathcal{A})$, questions $X(\mathcal{Q})$ and users, $X(\mathcal{U})$. We denote one answer as $\mathbf{x}_a$ in answer feature space $X(\mathcal{A})$, one question as $\mathbf{x}_q$ in $X(\mathcal{Q})$ and one user as $\mathbf{x}_u$ in $X(\mathcal{U})$.

Table 13: Features Spaces: $X(\mathcal{Q}), X(\mathcal{A})$ and $X(\mathcal{U})$

Question Feature Space $X(\mathcal{Q})$	
Q: subject length	Number of words of question subject
Q: detail length	Number of words of question detail
Q: posting time	Date and time when the question was posted
Q: question stars	Number of stars received earned for this question
Q: number of answers	Number of answers received for this question
Answer Feature Space $X(\mathcal{A})$	
A: overlap	Words shared between question and answer
A: number of comments	Number of comments added by other participants
A: total thumbs up	Total number of thumb up votes for the answers
A: total thumbs down	Total number of negative votes for the answers
User Feature SPace $X(\mathcal{U})$	
U: total points	Total points earned over lifetime community
U: questions asked	Number of questions asked
U: questions resolved	Number of questions resolved
U: total answers	Number of posted answers
U: best answer	Number of answers that were selected as "best answer"
U: stars	Number of stars the user receive
U: thumbs up ratio	The ratio of thumbs up votes the user posted before
U: thumbs down ratio	The ratio of thumbs down votes the user posted before
U: indegree	number of other users whose questions are answered by the user
U: outdegree	number of other users who answer the questions posted by the user
U: hub score	the hub score of the user computed by HITS
U: authority score	the authority score of the user computed by HITS

6.2.3.2 Learning Answer and Question Quality and User Reputation Using Coupled Mutual Reinforcement

Given an answer a, a question q and a user u described by feature vectors $\mathbf{x}_a$, $\mathbf{x}_q$ and $\mathbf{x}_u$, let the probability of them being a good answer, good question, good asker or good answerer be $P(\mathbf{x}_a)$, $P(\mathbf{x}_q)$, $P_{\text{qst}}(\mathbf{x}_u)$ and $P_{\text{ans}}(\mathbf{x}_u)$, respectively. In the following,

we will describe a generic approach to learning all these probabilities following the same way. We use P to denote any of $P(\mathbf{x}_a)$, $P(\mathbf{x}_q)$, $P_{\text{qst}}(\mathbf{x}_u)$ or $P_{\text{ans}}(\mathbf{x}_u)$ and use $\mathbf{x}$ to represent the corresponding feature vector.

Using logistic regression, we model the log-odds of $P(\mathbf{x})$ by the following linear models:

$$\log \frac{P(\mathbf{x})}{1 - P(\mathbf{x})} = \beta^T \mathbf{x} \tag{29}$$

where β are coefficients of the linear models. When given sufficient labeled instances, one can compute those coefficients by maximizing the corresponding log-likelihoods, say $LL(\mathcal{X})$ for Equation 29:

$$LL(\mathcal{X}) = \sum_{\mathbf{x} \in \mathcal{X}} y\beta^T \mathbf{x} - \log(1 + e^{\beta^T \mathbf{x}}) \tag{30}$$

where $y \in \{0, 1\}$ are the label of instance vector $\mathbf{x}$; $\mathcal{X}$ denotes the any of $\mathcal{U}$, $\mathcal{Q}$ or $\mathcal{A}$, which corresponds to the type of instance $\mathbf{x}$.

We can see that the above learning model depends exclusively on the corresponding feature space of the specific type of instances, i.e.,answers, questions or users. Thus the quality of questions and answers are learned only based on answer-related or question-related features while the reputation of users is estimated based on user-related features.

After adapting the coupled mutual reinforcement principle between the answer and question quality and the user reputation, showed in Equation 25, 26, 27 and 28, we are able to measure the conditional log-likelihood of observing one label set given some others belonging to different kinds of but associated entities. We use $\mathbf{y}$ to denote the current labels for $\mathbf{x}$ and use $\mathbf{y}'$ to denote new expected labels given the other kinds of old labels are known. We represent Y_d as the set of different types of entity associated with $\mathbf{y}$. For instance, based on Equation 26, the set Y_d of answer entity $\mathbf{y}_a$ is $\{\mathbf{y}_u^a, \mathbf{y}_q\}$, and in Equation 27, the set Y_d of users for questions $\mathbf{y}_u^q$ is $\{\mathbf{y}_q\}$.

We use KL-divergence to measure the conditional log-likelihood of $\mathbf{y}$ given associated Y_d:

$$LL(\mathbf{y}|Y_d) = -\sum_{i=1}^{|\mathcal{X}|} \mathbf{y}(i) \log \frac{\mathbf{y}(i)}{\mathbf{y}'(i)} - (1 - \mathbf{y}(i)) \log \frac{1 - \mathbf{y}(i)}{1 - \mathbf{y}'(i)} \tag{31}$$

And now we can extend the objective function from the original log-likelihood in Eq 30 to the following:

$$L(\mathcal{X}) = LL(\mathcal{X}) + \sigma LL(\mathbf{y}|Y_d) \tag{32}$$

where σ is a prescribed weight parameter. This equation represents the combined log-likelihood for learning the probability of each type of entity. Note that the KL-divergence can be combined with the original log-likelihood naturally because both are log-likelihood measured on probability distributions so are of the same units.

6.2.3.3 Fitting Models–Algorithm

Next, we show how to fit the above models and how to solve the parameter estimation problem. The idea is to start with uniform distributions for $P(\mathbf{x}_a)$, $P(\mathbf{x}_q)$, $P_{\text{qst}}(\mathbf{x}_u)$ and $P_{\text{ans}}(\mathbf{x}_u)$, and then iteratively update them to increase the likelihood based on their coupled mutually reinforcing relationship. In the following, we will first describe a generic approach to fitting any of the four logistic regression models for $P(\mathbf{x}_a)$, $P(\mathbf{x}_q)$, $P_{\text{qst}}(\mathbf{x}_u)$ or $P_{\text{ans}}(\mathbf{x}_u)$ in the mutual reinforcement framework. Then, we will describe an algorithm to learn answer and question quality and question-reputation and answer-reputation of user simultaneously.

We now describe how to fit the logistic regression model in more detail. As an example, consider fitting the model for the answer-reputation of users. From the current answer quality $\mathbf{y}_a$, we use Equation 25 to calculate $\mathbf{y}_u^a$. Note that we will keep the given labeled scores to the corresponding users. Then, given user-related features $X(\mathcal{U})$ and $\mathbf{y}_u^a$, we use the Newton-Raphson update to compute the new β_u^a

(denoted as $\beta_u'^a$) for the logistic regression model:

$$\beta_u'^a = \beta_u^a - \left(\frac{\partial^2 L(\mathcal{U})}{\partial \beta_u^a \partial \beta_u^{aT}}\right)^{-1} \frac{\partial L(\mathcal{U})}{\partial \beta_u^a} \tag{33}$$

Let $\mathbf{X}_u$ denote the matrix of $\mathbf{x}_u$ values, $\mathbf{p}_u^a$ the vector of the fitted probabilities of users and $\mathbf{W}_u^a$ a diagonal matrix with element (i,i) equal to $p_u^a(\mathbf{x}_i)(1-p_u^a(\mathbf{x}_i))$, then the Newton-Raphson step is thus:

$$\beta_u'^a = \beta_u^a + (\mathbf{X}_u^T \mathbf{W}_u^a \mathbf{X}_u)^{-1} \mathbf{X}_u^T (\mathbf{y}_u^a - \mathbf{p}_u^a) \tag{34}$$

We rewrite this equation as:

$$\beta_u'^a = (\mathbf{X}_u^T \mathbf{W}_u^a \mathbf{X}_u)^{-1} \mathbf{X}_u^T \mathbf{W}_u^a \mathbf{z}_u^a \tag{35}$$

where $\mathbf{z}_u^a = \mathbf{X}_u \beta_u^a + \mathbf{W}_u^{a\,-1}(\mathbf{y}_u^a - \mathbf{p}_u^a)$ is the residual. Using the new value of β_u^a, we are able to calculate the new answer-reputation of users $\mathbf{y}_u'^a$.

Then we can apply the same approach to fit the logistic regression model for the answer and question quality and the question-reputation of users (denoted as β_a, β_q and β_u^q, respectively).

Based on the proposed method of fitting logistic regression models, we present the following algorithm **CQA-MR** (Alg. 3) for simultaneously learning answer and question quality and user ask and answer reputation, where the ***Forward*** phase carries out the mutual reinforcement from the left to the right while the ***Backward*** phase from the right to the left as shown in Figure 26. Since we generate combined log-likelihood for learning the probability of each type of entity, the mutually reinforcing approach of our algorithm should cause the successive estimates of content quality and user reputation to converge. We will empirically demonstrate convergence in Section 6.4.

In this section, we have defined the problem of calculating content quality and user reputation in CQA. Then, We present coupled mutual reinforcement framework and a semi-supervised regression-based approach to solve the problem. In the following

Algorithm 3: CQA-MR

input : questions, answers and users and their connection from CQA-network.
output: answer quality $\mathbf{y}_a$;
answer-reputation of user $\mathbf{y}_u^a$;
question quality $\mathbf{y}_q$;
question-reputation of user $\mathbf{y}_u^q$
Start with an initial guess, e.g. uniform values, for $\mathbf{y}_a$, $\mathbf{y}_u^a$, $\mathbf{y}_q$ and $\mathbf{y}_u^q$;
begin
  while $\mathbf{y}_a$, $\mathbf{y}_u^a$, $\mathbf{y}_q$, $\mathbf{y}_u^q$ *not converge* **do**
    Forward fit the logistic regression models and calculate new values for $\mathbf{y}_a$, $\mathbf{y}_q$ and $\mathbf{y}_u^q$ in sequence ;
    Backward fit the logistic regression models and calculate new values for $\mathbf{y}_q$, $\mathbf{y}_q$ and $\mathbf{y}_u^q$ in sequence
end

sections, we will setup and carry on a large scale evaluation on the framework and our new method.

6.3 Experimental Setup

This section presents our evaluation setup. First, we describe our dataset including corpus of questions, answers and the corresponding users. Then, we describe our evaluation metrics and some methods for computing answer quality and user reputation used for comparison in the experimental results reported in Section 6.4. We also describe several ranking methods to illustrate the effects of user reputation and answer and question quality on general QA-retrieval.

6.3.1 Data Collection

In the experiments, we use the same dataset of factoid questions from the TREC QA benchmarks and collected CQA dataset as described in Section 5.3. And, we use the same method to obtain the relevance judgment.

Data Labeling for Content Quality and User Reputation

We use a set of labels for good users and good answers from Yahoo! Answers directly. For some question threads in Yahoo! Answers, there is one "*best answer*" which is

selected by the asker. These "*best answers*" can be viewed as high-quality answers. In addition, Yahoo! Answers selects some users as "*top contributors*" based on those users' answering history. These "*top contributors*" can also be viewed as users with high answer-reputation. In our data, there are 4000 "*top contributors*" and 18000 "*best answers*".

In order to evaluate the effectiveness of our algorithm for calculating answer quality and answer-reputation of users, we utilize a portion of these labels for users and answers. The other labels are used for testing by comparing with corresponding results of **CQA-MR**. In our experiments, we will keep 3600 *top contributors*' labels and 16000 *best answers*' labels for training our model, and then use the rest 400 *top contributors* and 2000 *best answers* to test the performance of our algorithm for learning answer quality and user answer-reputation. More importantly, we will evaluate the improvements to search, as described next.

In order to evaluate the effectiveness of our algorithm for computing question quality, we manually label a portion of the data. We randomly chose 250 resolved questions from Yahoo! Answers website which have received at least 5 answers. Two annotators were given 150 questions with 50 in common, and asked to label the quality of those 250 questions independently. The instructions for this labeling task were to consider both question subject and detail when examining question quality, and to consider answers when there is difficulty to understand the question. Questions were labeled as "*good*", "*fair*" and "*bad*", according to special guidelines share by annotators.

Table 14 reports the agreement between the two raters on the 50 common questions. Since sometimes it is very hard to distinguish between '*good*' and '*fair*' questions, we also combined '*good*' with '*fair*' to form a binary labeling. Both agreements are reported in Table 14. As we can see that we can get moderate agreement for both methods. As we can see, the binary labeling results in higher agreement; hence, we

will use the binary "*Good*"/"*Bad*" labels to evaluate question quality.

Table 14: Inter-annotator agreement and Kappa for question quality

	3 categories	2 categories
Agreement	64%	80%
Kappa coefficient	0.41	0.46

6.3.2 Evaluation Metrics

We adapt the same three information retrieval metrics, MRR, Precision at K, and MAP, to evaluate the performance of the our algorithm for learning answer quality as well as the performance of general QA retrieval.

6.3.3 Methods Compared

We now describe the methods used to compute user reputation, which we use for our main task of improving CQA retrieval. Specifically, we compare the following methods:

- **Baseline**: users are ranked by "indegree" (number of answers posted), an effective baseline estimate of user authority in CQA according to reference [81].
- **HITS**: we calculate the user reputation based on HITS algorithm. Users are ranked based on their authority scores.
- **CQA-Supervised**: we classify users into those with "high" and "low" reputation using a supervised classifier, namely SVM (SMO implementation) , trained over the features in Table 13. Then user are ranked based on their reputation scores.
- **CQA-MR**: predict user reputation based on our mutual-reinforcement algorithm (Section 6.2.3).

Unfortunately, a direct experimental comparison with reference [3], which is most closely related to our work, is impossible as neither the dataset or the truth labels

used for the experiments in [3] are available. However, **CQA-Supervised** is a similar approach and uses similar features to those described in [3], thereby providing a realistic state-of-the-art content quality classifier comparable to reference [3].

Our main task is to improve CQA retrieval by incorporating content quality and user reputation. We compare the following ranking methods:

- **Baseline**: In this method, the answers are ranked by the score computed as the difference of thumbs-up votes and thumbs-down votes received for each answer. This ranking closely approximates the ranking obtained when a user clicks "order by votes" option on the Yahoo! Answers site. The detail of this method and how to compute MRR and MAP under this setting is discussed in [9].

- **GBrank**: In this method, we apply the ranking method proposed in our previous work [9], which did not include answer and question quality and user reputation into ranking function. This method has been showed in [82] to have better performance than many state-of-the-art supervised ranking methods, such as RankSVM.

- **GBrank-HITS**: In this method, we optimize GBrank by adding user reputation calculated by HITS algorithm as extra features for learning the ranking function.

- **GBrank-Supervised**: In this method, we first apply a supervised method (SVM) to learn the answer and question quality and user reputation based on their individual feature set independently. Then, we optimize GBrank by adding obtained quality and reputation as extra features for learning the ranking function.

- **GBrank-MR**: In this method, we optimize GBrank by adding answer and

question quality and user reputation calculated by **CQA-MR** as extra features for learning the ranking function.

Note that, **GBrank-MR** and **GBrank-Supervised**, we use the same set of labels in learning.

6.4 Experimental Results

In this section, we will present several large-scale experiments. These experiments are used to demonstrate that (1) the **CQA-MR** algorithm exhibits good convergence behavior; (2) **CQA-MR** is effective for computing the question quality; (3) the performance of general QA retrieval can be improved by incorporating predicted quality features calculated by **CQA-MR**; (4) user reputation from **CQA-MR** tends to be better than those computed by other state-of-the-art methods; (5) the amount of supervision in **CQA-MR** affects the quality of predicted quality features.

6.4.1 Predicting Content Quality and User Reputation

6.4.1.1 Mutual Reinforcement Convergence

We first perform training using Algorithm **CQA-MR** introduced above. We examine the convergence behavior of **CQA-MR** by calculating the log-likelihood function (Eq. 30) over the iterations. We find that the log-likelihood values increase and converge at around 40 iterations, when computing content quality and user reputation.

We also calculate the log-likelihood for different values of σ. We are able to find that the log-likelihood has much smaller values when σ is bigger, especially in the initial few iterations, which means that the conditional log-likelihood ($LL(\mathbf{y}_a|\mathbf{y}_u^a, \mathbf{y}_q)$ and $LL(\mathbf{y}_u^a|\mathbf{y}_a)$) is very small initially. Therefore, the difference in the labels between successive iterations is big, which implies labels' inconsistency, in the early stage of the algorithm. However, we can also find that when we take more than ten iterations, the log-likelihood is almost the same regardless of the σ values. Thus, at later stage of our algorithm, the log-likelihood values are more sensitive to the objective function of

the logistic regression while the labels remain consistent across iterations, stabilizing at around 30 iterations.

6.4.1.2 Predicting Answer-Reputation

We now compare the effectiveness of **CQA-MR** with other methods for predicting answer-reputation of users. For this task, we use the hold-out set of 3600 users, with 678 of them labeled as "top contributors". Figure 27 reports the fraction of "top contributor" users included in the top K users ranked by answer-reputation, for varying K. As we can see, **CQA-MR** exhibits significantly higher precision than **CQA-Supervised**, which, in turn outperforms **HITS** and the simple "in-degree" count baseline.

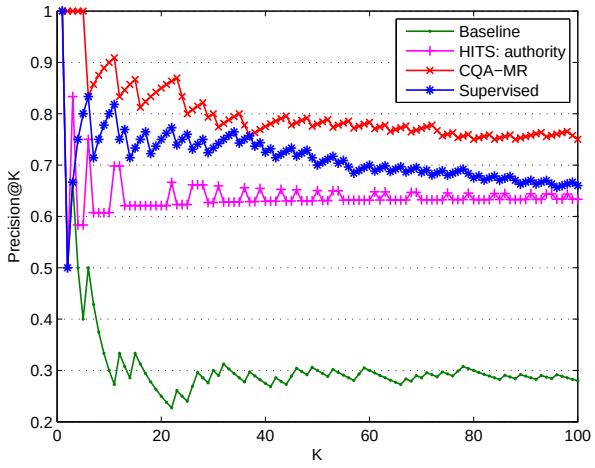


Figure 27: Precision at K for the status of *top contributors* in testing data

6.4.1.3 Predicting Question Quality

We now compare the effectiveness of **CQA-MR** with other methods for predicting question quality. For this task, we use the set of 250 questions with manually labeled quality, which is described in section 6.3.1. We train CQA-MR as described above (that is, no additional question quality labels provided for training) and randomly select 100 labeled questions for evaluating the performance of predicting question

quality. In order to compare with existing methods for predicting question quality, we also apply a supervised classifier, namely SVM (SMO implementation), trained over the features in Table 13. And the testing set is the same 100 labeled questions used above while the other 150 labeled questions are used for training SVM.

The mean average precision (MAP) for **CQA-MR** is 0.940 while that value is 0.890 for the supervised method (SVM). Figure 28 shows the precision-recall curves for both methods, it is clear that **CQA-MR** gives good performance on predicting question quality and exhibits significantly higher precision than supervised method. In addition, we also try to add 150 labeled question as seeds in training **CQA-MR**. Interestingly, adding the question labels as additional seeds for training CQA-MR does not significantly improve performance.

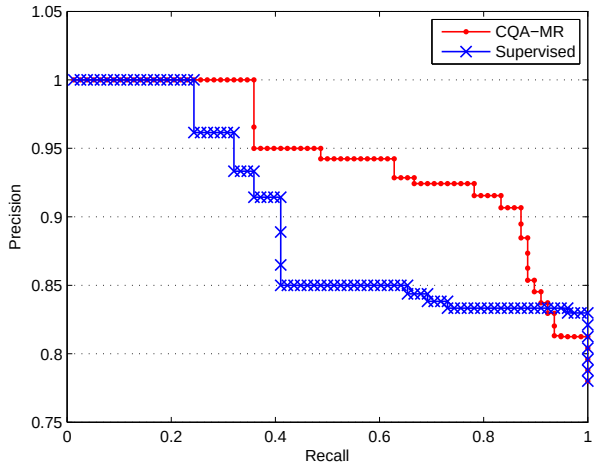


Figure 28: Precision-Recall curves for predicting question quality of **CQA-MR** and Supervised method.

6.4.2 Quality-aware CQA Retrieval

The answer and question quality and user reputation computed by **CQA-MR**, **CQA-Supervised** and **HITS** can be viewed as prior "*static*" features for QA retrieval since they are independent of queries. This complements "classical" information retrieval and QA retrieval, which primarily focused on query-dependent relevance features.

In this experiment, we seek to enhance the performance of general QA retrieval by incorporating predicted quality features (answer and question quality and user reputation). We use **GBrank** [82] as the ranking function and apply the same framework in our previous work [9]. For the training data we use 800 TREC queries and the associated community QA pairs, and we use another 450 queries and the associated community QA pairs for testing data. The set of features used to train the ranking function is described in detail in [9].

We train four ranking functions, **GBrank-MR**, **GBrank-Supervised**, **GBrank-HITS** and **GBrank**, on training data (i.e.,the 850 TREC queries) with predicted quality features added in the first three methods and training data without these features in the last one, respectively. Then, we test on the remainder hold-out testing data of 450 TREC queries and the associated community QA pairs.

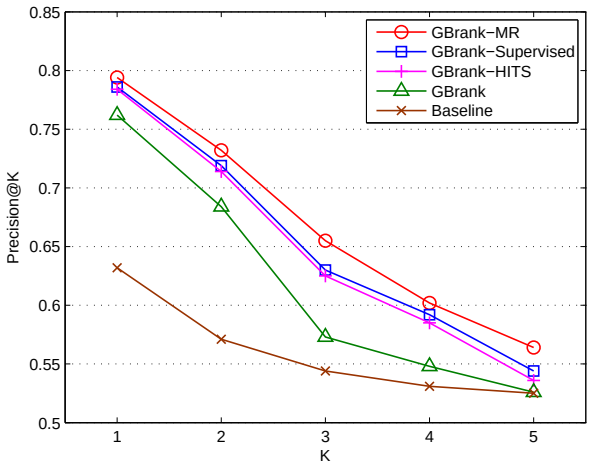


Figure 29: Precision at K for Baseline, GBrank, GBrank-HITS, GBrank-Supervised and GBrank-MR for various K

Figure 29 illustrates the Precision at K of **GBrank-MR**, **GBrank-Supervised** and **GBrank-HITS** compared with **GBrank** and the baseline method. The figure shows that all of the four ranking functions out-perform the baseline method. Furthermore, after incorporating predicted quality features, i.e.,answer and question quality and user reputation, **GBrank-MR**, **GBrank-Supervised** and **GBrank-HITS** give

better precision than **GBrank** without these features. In particular, the Precision at 1 of **GBrank-MR** is about 79% compared to 76% Precision at 1 exhibited by **GBrank**.

Table 15: Accuracy of GBRank-MR, GBRank-Supervised, GBRank-HITS, GBRank, and Baseline (TREC 1999-2006 questions)

	MRR	Gain	MAP	Gain
Baseline	0.664	-	0.443	-
GBrank	0.782	-	0.465	-
GBrank-HITS	0.827	+0.045(6%)	0.473	+0.008(2%)
GBrank-Supervised	0.833	+0.051(7%)	0.477	+0.012(3%)
GBrank-MR	**0.865**	+0.083(11%)	**0.483**	+0.018(4%)

Table 15 reports the MAP and MRR scores for **GBrank-MR**, **GBrank-Supervised**, **GBrank-HITS**, **GBrank** as well as the baseline method. Table 15 indicates that **GBrank-MR**, **GBrank-Supervised** and **GBrank-HITS** achieve much better performance than **GBrank** and the baseline method. In particular, for MRR scores, **GBrank-MR** achieves a gain of about 11% relative to **GBrank**; and **GBRank-MR** obtains double the gains of **GBrank-HITS** for both MRR and MAP scores. The above experiments illustrate the usefulness of the extracted static features in improving answer relevance.

6.4.3 Effects of the QA quality and User Reputation Features

We now explore the effects of the QA quality and user reputation, which is calculated by **CQA-MR**, on learning the ranking function. To this end, we perform a study on its influence on QA retrieval compared with existing graph-based method and supervised learning method for computing user reputation, i.e.,**HITS** and **CQA-Supervised**.

Figure 29 demonstrates the Precision at K of **GBrank-MR** compared with methods of **GBrank-HITS** and **GBrank-Supervised**. The **GBrank-Supervised** method replaces QA quality and user reputation calculated by a supervised learning

method. Note that we use the same set of labels in learning for **GBrank-MR** and **GBrank-Supervised**. In order to compare the two methods **GBrank-MR** and **GBrank-Supervised**, we apply t-test based on their precision and the p-value of significance test is 0.02. The **GBrank-HITS** method replaces user reputation by those calculated by the **HITS** algorithm.

Figure 29 and Table 15 indicate that **GBrank-MR** achieves much better performance than **GBrank-HITS**, which implies that the user reputation calculated by **CQA-MR** gives more contribution than user's authority scores computed by HITS. However, **GBrank-HITS** outperforms **GBrank** which does not contain QA quality and user reputation features. It shows that user's authority scores from **HITS** are still useful to enhance the performance of QA retrieval. Our conjecture is that for user's answer-reputation, it is much more important because **CQA-MR** not only utilizes network relationship but also individual reputation-related features while the authority scores in **HITS** only relies on the graph structure of CQA systems.

From Figure 29 and Table 15, we can also find that **GBrank-MR** performs significantly better than **GBrank-Supervised** ($p < 0.03$). After analyzing information gain of features, we find that **GBrank-MR** assigns higher weights on QA quality and user reputation features. All of these imply that the QA quality and user reputation calculated by **CQA-MR** gives more contribution than those calculated by supervised method with limited amount of training data. **GBrank-Supervised** also outperforms **GBrank** which shows that QA quality and user reputation obtained by supervised method are still useful to enhance the performance of QA retrieval.

6.4.4 Effects of the Amount of Supervision

As mentioned before, we utilize a set of training labels for users and answers in the algorithms **CQA-MR** and **CQA-Supervised** to learn predicted quality features. In this experiment, we show the influence of the amount of training labels (i.e.,degree of

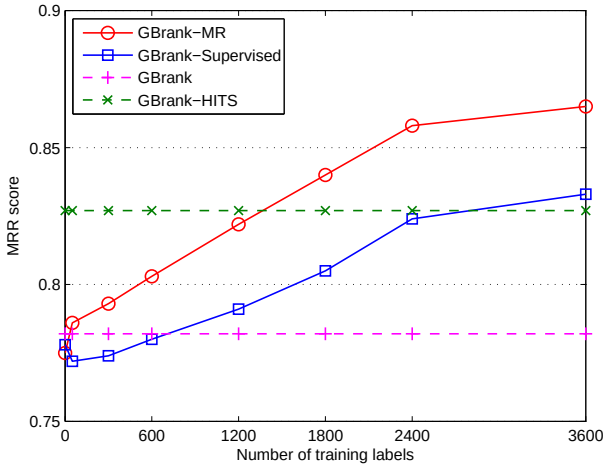


Figure 30: MRR of GBrank, GRrank-HITS, GBrank-MR, and GBrank-Supervised for varying fraction of training labels used in CQA-MR

supervision) on the performance of **CQA-MR** and **CQA-Supervised**. The labeled set contains 3600 *good users* and 16000 *good answers*. We vary the size of training labeled set used in the learning process by randomly selecting a certain number of labels. For example, choosing 40% labels means to use 2400 *good users* and 10800 *good answers* in the learning process.

Figures 30 and 31 report the MRR and MAP scores for the hold-out validation data against varying amount of labeled training data for high quality question and answer retrieval. We can see that MRR and MAP scores increase when there are more labels in **CQA-MR**. **CQA-MR** can achieve same accuracy as **CQA-Supervised** with about half of the required training data. Therefore, **CQA-MR** can improve QA retrieval much more with less supervised learning compared to **CQA-Supervised**. We also find that **GBrank-HITS** have higher accuracy than **GBrank-MR** when the amount of supervision is less than 1200 examples, suggesting that **HITS** indeed identifies high quality content/users, but can be improved on by our method.

In summary, our experimental results show that **CQA-MR** is an effective method for identifying high quality content and highly-reputable users in CQA, particularly when training data is limited. More importantly, we have shown that the predicted

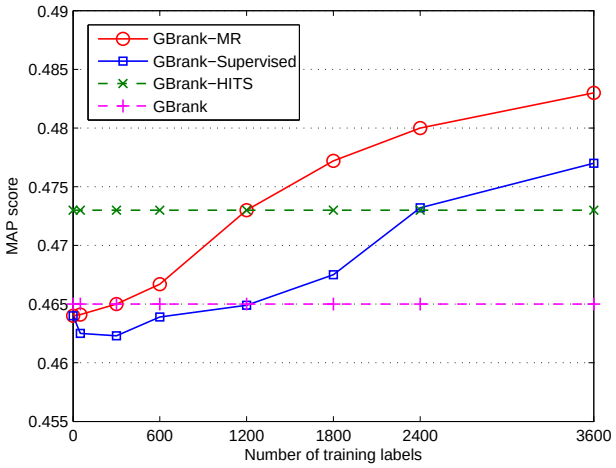


Figure 31: MAP of GBrank, GRrank-HITS, GBrank-MR, GBrank-Supervised for varying fraction of training labels used in CQA-MR

quality and reputation features, when modeled explicitly, significantly improve the accuracy of CQA retrieval. Finally, our experiments show the effectiveness of our mutual reinforcement algorithm, as **GBRank-MR** significantly outperforms a state-of-the-art supervised method (**GBRank-Supervised**, implemented using SVM) operating on the same features.

6.5 Summary

We presented CQA-MR, a framework for semi-supervised quality and reputation estimation of content and users in Community Question Answering. We have demonstrated the effectiveness of CQA-MR in large-scale experiments of a CQA dataset comprising over 100,000 users, 27,000 questions, and more than 200,000 answers. Our experiments demonstrate significant improvements over the supervised method, both in accuracy and in reducing the required amount of training data. Interestingly, for the task of predicting question quality, CQA-MR significantly outperforms a supervised method while requiring *no* question quality training labels. Furthermore, we demonstrated a significant improvement that CQA-MR provides for the practical task of searching the CQA archives.

As discussed in Chapter 5, beyond content quality and user reputation, the user feedback, such as voting, is another crucial context information for estimating the ranking relevance. However, not all the user votes are reliable. In the next chapter, we consider how to modify our ranking approach to make it more resilient to some common forms of vote spam.

CHAPTER VII

ROBUST RANKING IN SOCIAL MEDIA

7.1 User Votes in Social Media

Social media sources provide an effective alternative to traditional web search by directly connecting users with the information needs to users willing to share the information. For example, users can post questions or news items, and rely on other users to comment or rank the content (e.g., sites such as Slashdot or Digg). While the responses could be excellent, the quality could vary greatly. Hence, user feedback, such as voting, or rating the content, has become a crucial aspect of the effectiveness of the community. For example, in community question answering (e.g. Yahoo! Answers), users can give thumbs up or down votes to existing answers; while in social news and videos sharing services such as Digg [1] and Youtube, votes are used to judge the quality of the posted news or videos, as well for the quality of the comments. Figure 32 illustrates social content and user votes in the social media service.

Not surprisingly, user votes can provide crucial indicators into the quality and reliability of the content. For example, in Yahoo! Answers, more than half of the so called "best answers" to a question are chosen as the most popular answers according to the user votes. Our recent work [9] showed that incorporating user vote information can significantly improve the quality of a ranker over the CQA archives.

Unfortunately, not all user votes are reliable. Many "thumbs up" or "thumbs down" votes are generated without much thought, and, in some cases, by users intending to game the system – i.e., to promote specific answers or questions for fun or profit. We refer those bad or fraudulent votes as *vote spam*. We posit that vote spam

[1] http://digg.org/

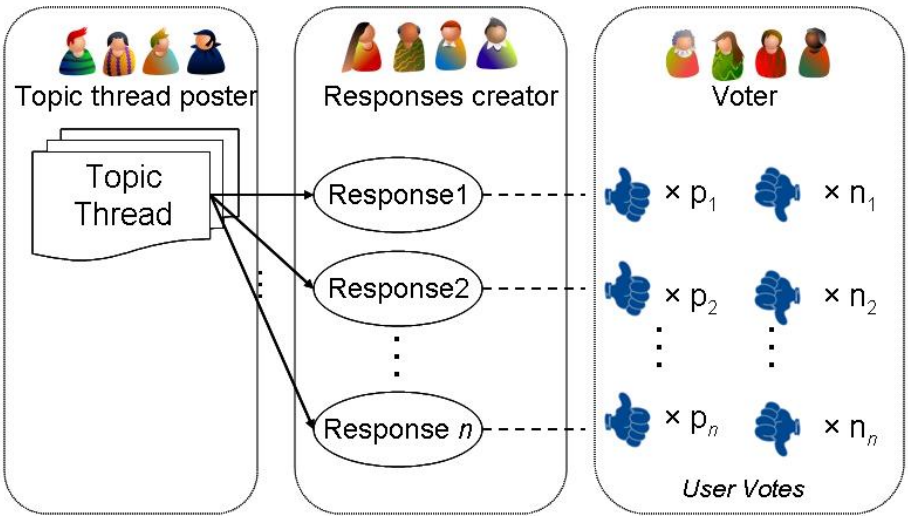


Figure 32: Illustration of social content and user votes in social media service: Users can post topic threads on social media sites **Topic thread poster**; Users can also submit responses to topic threads **Response creator**; Many social media services allow users to vote for existing responses using "thumb up" or "thumb down".

is an increasingly common phenomenon in social media sites, and deserves explicit handling for robust ranking of social media content. In the specific case of Yahoo! Answers, the support team already semi-automatically removes some of the more obvious vote spam after the fact. We believe that this solution may not prove adequate in the long run: as the amount and the patterns of vote spam evolve, post-factum filtering could significantly degrade user experience until the product team had a chance to react to a new spam attack. To complicate this problem, vote spam methods can change significantly due to varying popularity of content, specifics of media and topic and as spammers adjust their methods. Therefore, we need a robust method to train a ranking function that remains resilient to evolving vote spam attacks.

In this chapter we consider how to modify a recently presented algorithm for ranking social media [9] to make it more resilient to some common forms of vote spam. In order to investigate the robustness of our method, we consider several common vote spam methods in social media. In particular, we focus on the specific case of CQA to explore the influence of vote spam on the quality of answer retrieval.

7.2 Vote Spam in Social Media

As discussed before, user votes are valuable to evaluate the quality of user responses related to topic threads. In our consideration, there are two main types of vote spam in social media: incorrect votes and malicious votes. The user who gives the votes may not be an expert to the topic thread and related responses, therefore it is likely that its votes are incorrect. In another case, some malicious users intend to promote some specific responses within the community of social media, and they attack the social media service by creating a number of thumbs up vote to the specific responses. For example, in order to do online advertising in social media services, malicious users post their advertisement into responses to some topic threads and promote those responses by introducing amount of thumbs up votes. In another way, they can submit thumbs down votes to decrease the rank of high quality responses.

7.2.1 Vote Spam Attack Models

In this paper, we will focus on the influence of malicious vote attack and analyzing the robustness of our ranking framework. In the rest of this section, we will introduce a general vote spam model in the social media service.

We simulate the vote spam attack as following: Given the whole set of topic threads is $TP = \{tp_1, tp_2, \cdots, tp_m\}$, we assume that $\beta\%$ of them are attacked by malicious votes. As the goal of vote spam is to promote advertisement or other information from malicious users, those attackers tend to post and promote some the specific responses under popular topic threads. Thus, if a topic thread is more popular, i.e. followed by much more responses, it is more likely to be attacked. In this paper, the probability that a topic thread is included in $\beta\%$ attacked thread set is proportional to the number of its responses.

As the set of topic threads to be attacked has been selected, the number of attackers to each topic thread may be different. In our approach, we assume Gaussian

distribution to simulate the number of malicious users for each topic thread. We use N_i to denote the number of attackers to the topic thread tp_i, then

$$N_i \sim \mathcal{N}(\mu, \sigma^2)$$

Note that we can describe various number of attackers by simply changing the value of μ in Gaussian distribution. Other methods can also be used to simulate the number of malicious users.

For each topic thread to be attacked, we consider two general attack strategies, *thumbs up votes spam* and *thumbs up&down votes spam*. For *thumbs up votes spam*, malicious users aim to promote single response for one topic thread, so that they will propose thumbs up votes to the specific responses as many as possible. Using *thumbs up&down votes spam*, malicious users will give thumbs down votes to other responses in addition to thumbs up vote for one specific responses. In this way, attackers can promote specific responses by decreasing the ranking of others.

The next question is how many thumbs up or down votes malicious users will "contribute" for a particular topic. Most community portals and social media services enforce strict budget rules for user votes which constrain the number of votes one user can give. In our simulated attack approach, we assume an unlimited overall voting budget for a user, but include a common restriction that any user can vote (thumbs up or down) at most once for each item.

Figure 33 summarizes our attack models: First, we choose $\beta\%$ topic threads to attack; Then, the number of attackers for each attacked thread is decided based on Gaussian distribution; After selecting one response to promote for each attacked thread, we can choose one attack strategy which is either *plus vote spam* or *plus-minus vote spam*. In the next section, we will study Community Question Answering(CQA) service. We use the proposed framework for learning ranking function for QA retrieval as well as evaluate the robustness of learned ranking against vote spam.

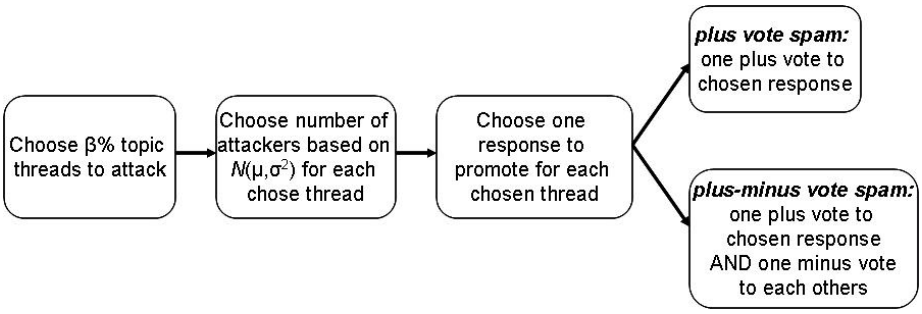


Figure 33: Summary of the stochastic vote spam generation process.

7.3 Robust Ranking in Community Question Answering

7.3.1 Robust Ranking Method

As shown in our previous work [9], our ranking function (*GBrank*) exhibits promising performance on QA retrieval. Our experiments demonstrate that user vote information provides much contribution to the high accuracy of our *GBrank*, when there is no vote spam. However, if user votes in CQA have been polluted by spam from malicious users and we continue using *GBrank* trained by clear data without vote spam, *GBrank* will still put much reliance on user vote information which however is supplying inaccurate information due to the spam.

In order to create a robust ranking method, we enhance our *GBrank* by using polluted training data during learning process. We apply the general vote spam model, described in Section 7.2, to generate vote spam into unpolluted QA data. Then, we train the ranking function based on new polluted data. To distinguish with *GBrank* trained by clear data, we denote our new ranking function as *GBrank-robust*. *GBrank-robust* is able to automatically account for the observed noise in the preference features, transferring more weight to other content and community features. Therefore, *GBrank-robust* can outperform the original *GBrank* method where simulated vote spam was not introduced at training time.

In the rest of this section, we will present our evaluation setup. First we describe our dataset including the queries and the corresponding corpus of questions and

answers. Then we describe the ranking methods to compare for the experimental results reported in Section 7.4.

7.3.2 Datasets

To evaluate our proposed robust ranking method, we use the same dataset as described in Section 5.3.1, which consists of factoid questions from the TREC QA benchmarks, question-answer collection from Yahoo! Answers, and automatically generated relevance judgments.

Vote Spam Recall that there may already be some spam in the original data, but it is limited as we are experimenting with past data that has been cleaned to large extent by the Yahoo! team. Therefore, we need to inject vote spam by ourselves. The model of vote spam has been discussed in Section 7.2. We will describe the parameters of the settings of vote spam in Section 7.3.4.

7.3.3 Ranking Methods Compared

To evaluate the QA retrieval quality, we compare the quality of following methods:

- **Baseline**: In this method, the answers are ranked by the score computed as the difference of positive votes and negative votes received for each answer. This ranking closely approximates the ranking obtained when a user clicks "Order by votes" option on the Yahoo! Answers site.

- **GBrank**: Ranking function with textual and community/social features: this is our method presented in Section 5.2.4.

- **GBrank-robust**: Similar to GBrank, we utilize textual and community features to train ranking function. However, the training data is polluted according to the chosen spam model. We will discuss how to evaluate vote spam's influence on QA retrieval in Section 7.3.4.

7.3.4 Evaluation on Robustness of Ranking to Vote Spam Attack

To evaluate the robustness of proposed QA ranking algorithm, we compare the performance of the ranking algorithm in the situation with vote spam and without spam. In detail, we consider the following settings of vote spam and evaluate their influence on performance of ranking respectively:

- *Scope of Vote Spam*: The scope of vote spam is measured by the percentage of attacked question threads (β). We will compare the performance of ranking under vote spam attack when the number of attacked topic threads ($\beta\%$) is of different value.

- *Number of Attackers*: We will compare the performance of ranking under vote spam attack when the number of attacker varies. In our paper, the number of attacker for each question thread obey Gaussian distribution(Section 7.2.1). In this paper, we fix the variance σ^2 in Gaussian distribution and model the number of attackers by using different mean μ.

- *Attack Strategy*: We will compare the performance of ranking under two different strategies of vote spam attack: *thumbs up vote spam* and *thumbs up&down vote spam*. In this first strategy, malicious users promote one specific answer only by adding thumbs up votes to it. In the second one, attackers submit not only thumbs up votes to the specific answer but also thumbs down votes to the other answers in the same question thread.

7.4 Experimental Results

7.4.1 QA Retrieval

In this experiment, we investigate the performance of QA retrieval under the attack of vote spam.

In order to simulate vote spam, we use the method described is Section 7.2.1 to add vote spam in dataset: we first sample 10% question threads to be attacked, denoted as $\{q_1, q_2, \cdots, q_s\}$. Then, we assume the number of attackers obeys the Gaussian distribution whose mean is 3 and variance is 1 and sample attacker numbers for each attacked thread, denoted as $\{N_1, N_2, \cdots, N_s\}$ respectively. In each sampled question thread q_i, we randomly select one answer which will be promoted by malicious users. And we use *thumbs up vote spam* strategy for attacking, i.e. adding N_i thumbs up votes to the specific answer in q_i respectively. In the following experiment, we use this attack model by default.

In our experiment, we train two ranking functions, *GBrank-robust* and *GBrank*, on training data (i.e., the 800 TREC queries) with vote spam and training data without vote spam respectively. The remainder hold-out testing data (i.e. 450 TREC queries and associated community QA pairs) is added with vote spam. Then, using the polluted testing data, we evaluate the performance for two ranking functions and baseline method.

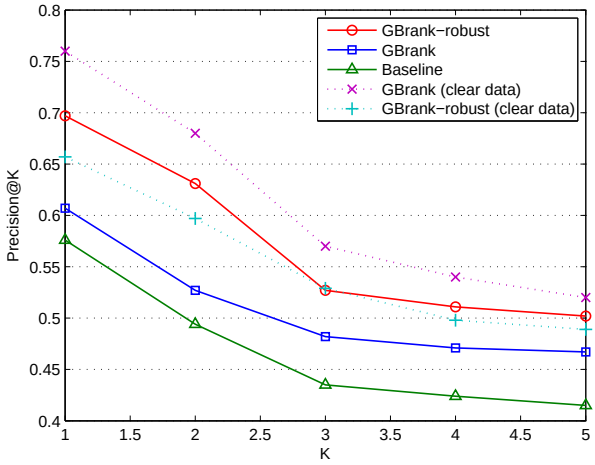


Figure 34: Precision at K for Baseline, GBrank and GBrank-robust for various K.

Figure 34 illustrate the Precision at K of *GBrank* and *GBrank-robust* compared with the baseline method. Testing data for all the three metrics are extracted from

polluted dataset. (Vote spam model: $\beta\% = 10\%$, $\mu = 3$ and $\sigma^2 = 1$). This figure shows that, under the situation that test data is polluted with vote spam, *GBrank-robust* still performs better than baseline method, but *GBrank* does not obtain better performance than baseline.

In Figure 34, we also demonstrate the Precision at K of *GBrank* and *GBrank-robust* when they are evaluated using unpolluted testing data. It is clear to see that *GBrank-robust* is "*graceful*" in that it is not much worse than original *GBrank* when there is actually no vote spam.

Table 16: Mean Reciprocal Rank (MRR) and Mean Average Precision (MAP) for GBrank, GBrank-robust and Baseline

	MRR	MAP
GBrank	0.648	0.416
Baseline	0.624	0.405
GBrank-robust	**0.736**	**0.457**

In Table 16, we illustrate the MAP and MRR scores for the baseline method as well as *GBrank* and *GBrank-robust*. From the table, it is clear that *GBrank-robust* reaches much better performance than *GBrank* and baseline method. In particular, *GBrank-robust* still achieves a gain of about 15% relative to the baseline.

To understand how two GBrank functions can outperform an "oracle" baseline, consider that the ordering of answers within a question thread remains fixed (either by date – as the default, or by decreasing votes). In contrast, GBrank obtains a better ranking of answers within each question thread, as well as a global ranking of all answers. Then, improved ranking within each Yahoo questions thread contributes to the higher score than baseline. Overall, applied on Yahoo! Answers, our proposed framework achieves a significant improvement on the performance of QA retrieval over the Yahoo! Answers' default ranking and the supported optional votes-based ranking. In addition, from the experiment, we can find that our method is able to retrieve

relevant answers at the top of results. In summary, we have shown that *GBrank-robust* significantly outperforms extremely strong baselines, achieving precision at 1 of nearly 70% and MRR of over 0.73, which are high values even for traditional QA retrieval.

7.4.2 Robustness to Vote Spam

In this section, we perform experiments to evaluate the robustness of our ranking function to user vote spam. As discussed in Section 7.2.1, there are three parameters to decide the model vote spam attack: scope of vote spam, number of attackers and attack strategy. We will also assess the vote spam influence on our ranking function under various settings of attack model.

In the following, we illustrate the influence of vote spam on *GBrank-robust* under various settings of attack model. Note that *GBrank-robust* is trained using default vote spam model ($\beta\% = 10\%$, $\mu = 3$ and $\sigma^2 = 1$). In particular, we evaluate our model's sensitivity to various parameter settings by using the testing data polluted by different vote spam model.

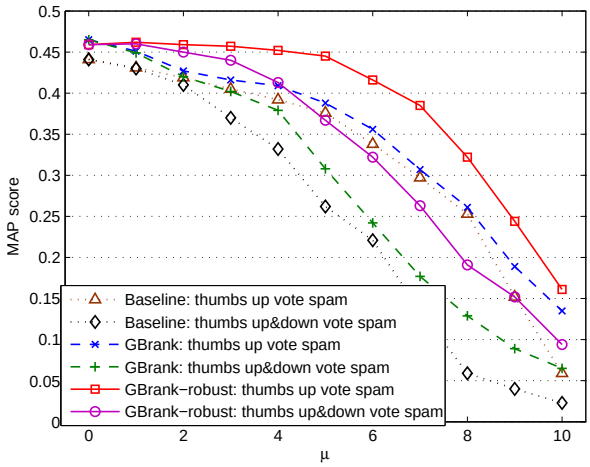


Figure 35: MAP scores for GBrank-robust, GBrank and Baseline for various mean number of attackers. We calculate the scores for thumbs up vote spam and thumbs up&down vote spam respectively.

1. Number of attackers

First, we explore whether the number of attackers for each question thread affects *GBrank-robust* performance. As discussed above, we model number of attackers for each thread as Gaussian distribution($\mathcal{N}(\mu, \sigma^2)$). In our experiment, we fix σ^2 to 1 and only use *thumbs up vote spam* for the testing data. The other parameters remain default values shown above. Figure 35 illustrate the MAP scores of *GBrank-robust*, GBrank and baseline method under vote spam with different number of attackers for the test data, i.e. various value of μ. These figures shows that the performance of *GBrank-robust* outperform than both GBrank and baseline when there are vote spam in testing data. Although it declines as the average number of attackers increases, *GBrank-robust*, as we can see, is more robust to the vote spam.

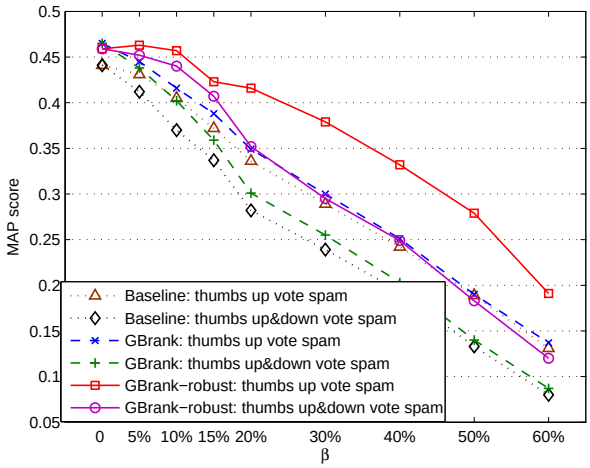


Figure 36: MAP scores for GBrank-robust and Baseline for various scope of vote spam. We calculate the scores for thumbs up vote spam and thumbs up&down vote spam respectively.

2. Scope of vote spam

Second, we investigate whether the scope of vote spam influences *GBrank-robust* performance. In our experiment, we use both *thumbs up vote spam* and *thumbs up&down vote spam* strategies and the other parameters remain default values except

percentage of attacked question threads. Figure 36 illustrate MAP scores of *GBrank-robust* on the testing data which are polluted by different scope of vote spam, i.e. various value of $\beta\%$. It is showed that the performance of *GBrank-robust* outperform than both GBrank and baseline when there are vote spam in testing data. Although it declines while the scope of vote spam arises, *GBrank-robust*, as we can see, is more robust to the vote spam.

3. Attack Strategy

In addition, to gain understanding of how different spam strategies achieve different decreasing on performance, we perform the same experiment and illustrate the results in Figure 35 and 36. From these figures, it is obvious to see that *thumbs up&down vote spam* can cause more serious loss on ranking performance than *thumbs up vote spam*. We consider the reason behind is that *thumbs up&down vote spam* probably disorder much more the correct preference as this strategy gives more vote spam than *thumbs up vote spam*.

7.4.3 Analyzing Feature Contributions

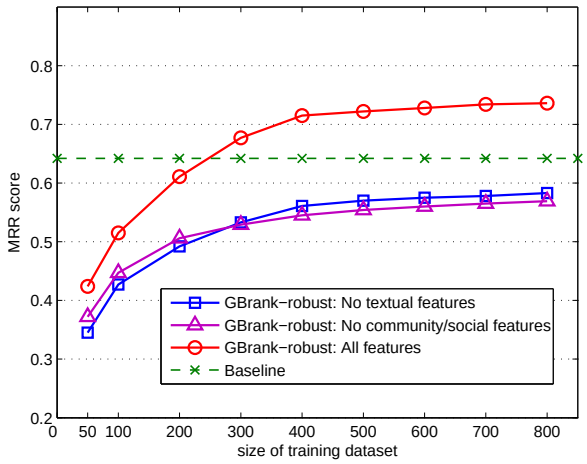


Figure 37: Mean Reciprocal Rank(MRR) for GBrank-robust with different size of training data on feature ablation study. The MRR for baseline using polluted data is also shown in the figure.

To gain a better understanding of the important features for this domain, we carry out an ablation study when training and testing data have been polluted by vote spam. Figure 37 illustrates the MRR scores when learning ranking function with all features or removing one of categories, and the learning is based on various size of polluted training data. The MRR score of baseline method is presented in the figure as well. From the figure, we can find that removing either textual features or community features cause a significant degradation of performance, especially for small amounts of training data. Even in the presence of vote spam, our ranker with all of the features is able to outperform the baseline after only 300 questions in the training set. In summary, we have shown that our ranker is both effective and robust to a variety of vote spam methods.

We evaluate the feature contributions for all the features both when training and testing data are polluted by spam and when neither is polluted. Table 18 shows the Information Gain for all features when training and testing data have been polluted by vote spam; While Table 17 shows the Information Gain for all features when training and testing data have been polluted by vote spam.

Table 17: Information Gain for all features both when training and testing data are not polluted

Info Gain	Feature Name
0.048	similarity between query and question
0.045	number of resolved question for answerer
0.043	length ratio between query and answer
0.032	number of thumbs down vote
0.030	number of stars for answerer
0.021	number of thumbs up vote
0.014	similarity between query and qst + ans
0.013	number of answer terms
0.013	number of question asked by answerer
0.011	answer's lifetime

From these two tables, we can find that (1) some textual features and community features have much influence on the ranking function. (2) When there is no spam

Table 18: Information Gain for all features both when training and testing data are polluted

Info Gain	Feature Name
0.048	similarity between query and question
0.045	number of resolved question for answerer
0.043	length ratio between query and answer
0.029	number of stars for answerer
0.026	similarity between query and qst + ans
0.018	number of answer terms
0.013	number of question asked by answerer
0.011	answer's lifetime
0.010	number of question terms
0.009	length ratio between query and question
...	...
0.003	number of thumbs down vote
0.002	number of thumbs up vote
...	...

in training and testing data, user vote information is important to ranking results. (3) However when incorporating vote spam in training and testing data, both of them contribute much less than before. Therefore, vote spam can give rise to a little decreasing on the performance of ranking function; however, due to the contribution of textual features and community features, our ranking function works in a robust way.

Based on our experiments reported in this section, we conclude that *GBrank-robust* is resilient to vote spam under various settings of the vote spam attack model. And even when the test data does not have overt vote spam, *GBrank-robust* degrades only slightly over our original *GBrank* method (and still performs significantly better than a state-of-the-art baseline). We also observe that *GBrank-robust* is not sensitive to the specific parameters of the vote spam model. Our feature analysis shows that *GBrank-robust* manages to automatically assign more weight to the textual and other more difficult-to-spam interaction features when properly trained.

7.5 Summary

Social media is transforming the way people find and evaluate information online. Users do not only share information on social media sites, but also contribute their ratings of the content. As such, user feedback has become a crucial mechanism for content quality control, ranking, and filtering. Unfortunately, as the popularity of community and social media sites grows, so do the incentives and incidents of malicious user behavior, such as vote spam.

We have presented a robust, effective method which incorporates social and content information for retrieving information from social media. In particular, we focused on the *robustness* of ranking in the presence of malicious feedback (vote spam), analyzing general models for common vote spam strategies and developing a training method that improves the robustness of ranking by injecting simulated spam into the training data. Our extensive experiments on a particularly important case of social media –Community Question Answering– demonstrated the effectiveness of our approach, which is robust in that the accuracy of ranking degrades gracefully with increased spam, better than unpolluted model. Furthermore, we have shown that our method of training is resilient to vote spam that is significantly different from the training data.

CHAPTER VIII

LEARNING TO ORGANIZE THE KNOWLEDGE IN SOCIAL MEDIA

Followed by our research on ranking over social media, this chapter investigates techniques for integrating and organizing Web content in social media. Specifically, we study how to organize the individual task questions in CQA into structured knowledge, which can be considered as new form of context information. Using the extracted structured semantics of task questions, we are able to improve the performance of external applications, such as question semantics identification and similar question retrieval.

8.1 Task Question and Its Structured Semantics

Task question represents a specific type of information need, in the form of natural language question, which usually implies that the user intends to accomplish a certain task and thus is searching instructions related to some aspects of the task. For example, "*how to find the cheapest hotel in New York City?*" is a task question posted by a Web user. The asker apparently is planning a task of "*travel to New York City*", and he needs the information about the aspect of "*hotel*" for this travel task.

Recently, task questions have been widely used by normal Web users to describe their information needs during Web search, especially when they need some instructions from outside to help them make the decision on the tasks. However, finding relevant instructions for task questions is a difficult mission that is distinct from retrieving relevant Web documents for generic Web search queries. Firstly, compared

with generic search queries in the form of bag of words, task questions can represent more accurate semantics of the users' information needs in the form of natural language questions, which requires new effective methods to interpret the semantics of task questions. Secondly, the users asking task questions usually prefer to seeing an integrated and well-organized task-oriented knowledge as instructions rather than browsing a list of Web documents returned by generic search engine. This also requires effective methods to integrate information relevant to the task questions.

Community Question Answering (CQA), emerging as a popular forum for users to post questions and answer those questions, provides an alternative channel for solving task questions. Over the last few years, CQA portals have exploded in size and popularity. For example, Yahoo! Answers [1] already has stored hundreds of millions of questions and their associated answers, among which a large fraction of questions are task questions. There also exists a CQA portal, eHow [2], collecting only task questions, each containing the phrase "*how to*". However, most of questions in eHow are created by experts and the number of archived questions is much smaller than that of Yahoo! Answers. To ensure wide coverage, in this paper, we focus on studying task questions in Yahoo! Answers. Observing that task questions usually contain the phrase "*how to*", we simply consider questions having this phrase as task questions in CQA portals. As observed, there are about as many as tens of millions such kind of task questions. In addition, the number is consistently growing as Web users keep posting more questions to CQA portals.

Such a large number of task questions comprise a valuable knowledge repository, which could be a gold mine for automatic instruction searching and task solving. However, as discussed above, to make the immense body of knowledge easy-to-understand, effective methods are required to interpret the semantics of task questions and further

[1] http://answers.yahoo.com/
[2] http://www.ehow.com/

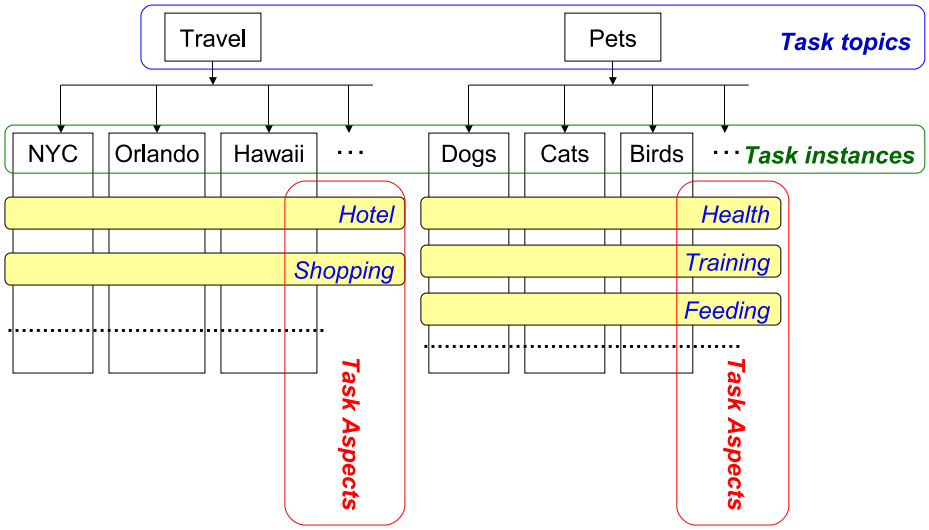


Figure 38: Structured semantics of task questions in CQA

integrate task questions into a well-organized knowledge structure. In this study, we address these challenges by extracting structured semantics of task questions.

Currently, CQA portals usually organize all questions into a hierarchical semantic classification, which can be utilized to identify structured semantics for task questions. For example, Figure 38 shows a part of the hierarchy, where the first layer classes, e.g. *travel*, *pet*, are considered as general **task topics**, and each task question under one task topic, say *travel*, is related to *how to do a task for traveling*; the second layer ones, e.g. *NYC* and *Orlando* under *travel*, are considered as **task instances** of the task topic, and each task question under one task instance, say *NYC*, is related to *how to do a task for traveling to NYC*. In CQA portals, the categorization of each task question is decided by the user who posted the question, and one task question belongs to only one task topic and one instance.

However, such existing question classification ignores another latent dimension of semantic structure: each task topic consists of several **task aspects**. As shown

in Figure 38, for the task topic *travel*, there are a set of aspects such as *hotel* and *shopping*. This aspect-based dimension can apparently provide more essential interpretation of the semantics of the task questions. Thus, for the goal of extracting structured semantics of one task question, it is necessary to recognize the task aspect that the task question corresponds to, in addition to identifying the task instance that the question belongs to.

In this paper, we formally define the structured semantics of each task question as a tuple $< T, I, A >$, consisting of a task topic T, a task instance I of topic T, and an aspect A of topic T. Based on such structured semantics, we are able to organize all task questions into integrated task summaries. This kind of integration is quite useful. Firstly, it automatically provides a hierarchical organization of all task questions which makes it more easily for users to browse the task knowledge in CQA. Secondly, such hierarchical knowledge can also be employed to effectively organize new arriving questions. Furthermore, integrated task summaries can benefit question-answering retrieval in terms of both relevance and diversity. Specifically, it can not only improve the performance of retrieving relevant questions for the new question, but also support finding questions which are under the same task topic as the new question but in different aspects.

As most CQA portals have provided hierarchical semantic classification, which can be used to extract task topics and instances, the most important problem becomes identifying the task aspects of task questions. Note that, under each task topic, there exist a set of task aspects that are common across all task instances, but are characterized uniquely for each task instance. For example, *travel to NYC* and *travel to Orlando* are two instances of the task topic *travel*, *hotel* is one of aspects of *travel*, thus both *travel to NYC* and *travel to Orlando* have the aspect of *hotel*, but, according archived questions in Yahoo! Answers, we observe that users tend to find *cheap hotels* when *traveling in NYC* while they are more interested in *hotels near the Disney*

when *traveling in Orlando*. This problem is considered as a comparative text mining (CTM) [80] problem and is challenging since (1) we need to identify task aspects across different task instances of the same task topic; (2) for each discovered aspect, we also want to identify the unique information specific to each task instance. In this paper, we address this problem by employing a generative probabilistic mixture model for CTM [80], which simultaneously performs both cross-instance and within-instance clustering, and can be applied to an arbitrary task topic including a set of comparable task instances. This method is quite general, since it can be used to extract semantics and generate integrated summary for any collection of social media content with semantic structure of topic-instance-aspect.

Furthermore, we hope to solve the problem with minimum supervision; however, if there exists expert knowledge on the structured semantics of task questions, we also want to incorporate this information into the model. For example, eHow has assigned questions into specific task instances and aspects according to experts' opinions, which could be considered as prior expert knowledge in the model. In this paper, we propose to cast such knowledge as a prior in the probabilistic mixture model and estimate the model to obtain the structured semantics aligned with expert knowledge as well as additional semantics not well-aligned with the expert knowledge.

The rest of this chapter is organized as follows. Section 8.2 formally defines the problem and propose our generative mixture model. After that, we introduce some applications of using the structured semantics of task questions in Section 8.3. Experimental setup and results are illustrated in Section 8.4.

8.2 Integrating Task Questions Based on Structured Semantics

8.2.1 Problem Formulation

Let $\mathcal{Q} = \{q_1, ..., q_{|\mathcal{Q}|}\}$ be a set of task questions archived in the CQA portal, and each question $q_i \in \mathcal{Q}$ is associated with one best answer a^{q_i}. We assume that $\mathcal{Q}$ covers

a number of general **task topics**; and each general task topic includes a number of **task instances**; moreover each task topic consists of a couple of **task aspects** shared across all instances under the task topic. Usually, we can map task questions into a hierarchical semantic classification to form up a *task-instance structure*, as many CQA portals have done.

Definition 5 (Task-Instance Structure)*: A* **task-instance structure** *is a two-level hierarchical semantic classification. The first level consists of a set of general task topics,* $\mathcal{T} = \{T_1, T_2, \cdots, T_{|\mathcal{T}|}\}$*, and the second level includes a set of task instances. Each task instance belongs to only one task topic, and each task topic* T_i *contains a number of task instances,* $\mathcal{I}(T_i) = \{I_1^i, I_2^i, \cdots, I_{|\mathcal{I}(T_i)|}^i\}$*.*

Following previous approaches on theme analysis, we represent general task topics as a set of theme models $\{\theta_1^T, \theta_2^T, \cdots, \theta_{|\mathcal{T}|}^T\}$, each being characterized by a multinomial distribution over all the words in our vocabulary (also known as unigram language model).

Definition 6 (Task Topic Model)*: A* **task topic model** θ^T *in a question collection* $\mathcal{Q}$*, is a probabilistic distribution of words* $\{p(w|\theta^T)\}_{w \in V}$ *and represents a semantically coherent general task topic. Clearly,* $\sum_{w \in V} p(w|\theta^T) = 1$*.*

The most important thing for extracting structured semantics of task questions is to identify aspects for each task topic. We assume that, under one task topic T, there are k common aspects (themes), $\{\theta_1^A, \cdots, \theta_k^A\}$; and for each instance $I_i \in \mathcal{I}(T)$, there are k instance-specific aspects (themes), $\{\theta_{1,i}^A, \cdots, \theta_{k,i}^A\}$. Each common aspect or instance-specific aspect is characterized by a multinomial distribution over all the words in our vocabulary.

Definition 7 (Common Aspect Model)*: A* **common aspect model** θ_j^A *is a probabilistic distribution of words* $\{p(w|\theta_j^A)\}_{w \in V}$ *and represents a common aspect of a specific task topic. Clearly, we have* $\sum_{w \in V} p(w|\theta_j^A) = 1$*.*

Definition 8 (Instance-specific Aspect Model): *An* **instance-specific aspect model** $\theta^A_{j,i}$ *is a probabilistic distribution of words* $\{p(w|\theta^A_{j,i})\}_{w\in V}$ *and denotes an instance-specific aspect of a task topic, where* $\sum_{w\in V} p(w|\theta^A_{j,i}) = 1$.

In this paper, we would like to extract *structured semantics* for each task question and generate the *integrated task summary* based on the question collection of each task topic.

Definition 9 (Structured Semantics): *The* **structured semantics** *of one task question* q *is a tuple* (T_q, I_q, A_q) *where* T_q, I_q *and* A_q *represent the task topic, task instance, and task aspects that the question* q *belongs to, respectively.*

Definition 10 (Integrated Task Summary): *An* **integrated task summary** *of a certain task topic* T *is a tuple* $(T, \mathcal{I}(T), \mathbf{A}(T))$, *where* $\mathcal{I}(T)$ *denotes the set of instances of the topic* T; $\mathbf{A}(T) = \{A^T_1, \cdots, A^T_k\}$ *represents the description of* k *aspects of the topic* T; *and for each aspect description,* $A^T_j = \{A^{T,C}_j; A^{T,1}_j, \cdots, A^{T,|\mathcal{I}(T)|}_j\}$ *where* $A^{T,C}_j$ *is the description of the* j*-th common aspect of the topic* T *and* $A^{T,i}_j$ *denotes the description of the* j*-th aspect of the topic* T *specified by the instance* I_i.

Note that there are two application scenarios: (1) *No supervision:* If there is no prior knowledge of the aspects, we just automatically extract purely ad hoc aspects based on the data; (2) *Minimum supervision:* If there exists a couple of keywords or even several example questions specifying either the common aspects or instance-specified ones from prior expert knowledge, we should align the extracted common or instance-specified aspects with the prior knowledge.

8.2.2 Aspect Discovery and Clustering on CQA Content

Since the topic-instance structure is usually provided explicitly by CQA portals, we focus on aspect discovery among the questions under each task topic. A simple solution to aspect discovery is to ignore the structure of multiple task instances and treat

all questions from different task instances as a single collection, where the standard clustering algorithm can be applied, such as k-means. Specifically, we could represent each question q as a vector of $v(q) = \langle c(w_1, q),\ c(w_2, q) \cdots \rangle$ where $c(w, q)$ denotes the number of word w occurring in q, and apply k-means to a set of such vectors. The hope is that the obtained clusters would represent the common aspects across all instances. However, the word space is usually of very high dimensionality in the context of social media content. Due to the *curse of dimensionally*, the sparsity of the data could affect the clustering performance.

To minimize the affect of data sparsity, we can employ the probabilistic topic model to perform clustering, such as Probabilistic Latent Semantic Analysis (PLSA) [28] and its extensions [54, 56], which have been successfully applied to many text mining problems with promising results. If we consider questions from different task instances as a single collection, we could apply PLSA to cluster these questions in order to discover task aspects.

As in most topic models, the general idea is to use a multinomial word distribution (i.e. unigram language model) to model a theme. Assuming k latent common aspects in all task instances, each corresponds to a theme model, a task question is regarded as a sample of a mixture model with these theme models as components. We can fit such a mixture model to the union of all the task instances, and the estimated component theme models can be used to analyze the common aspects and differences among the instances. Usually, we can apply an Expectation-Maximization (EM) algorithm to compute a maximum likelihood estimate for the theme models.

However, this simple model treats aspect discovery under a certain task topic as a single-collection text mining problem, without considering the difference and similarity between various task instances. Such model is inadequate for comparable text mining (CTM) for two reasons: (1) The task topic-instance structure is completely

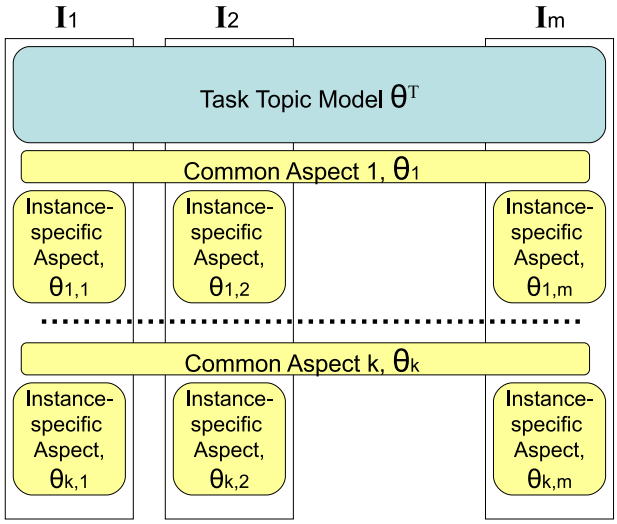


Figure 39: The cross-collection mixture model

ignored. As a result, we may have aspects that represent only some, not all of the instances; (2) It is uneasy to identify which aspect represents the common information across different task instances and which represents specific information to a particular instance. In the following, we will present a more sophisticated mixture model addressing the two deficiencies that is specifically designed for CTM.

8.2.2.1 CTM-PLSA

The main idea for improving the basic PLSA for CTM is to explicitly distinguish *common aspects* shared across all task instances from *instance-specific aspects*. Thus, we now consider k latent common aspects as well as a potentially different set of k instance-specific aspects for each task instance (as illustrated in Figure 39). These component models directly correspond to both the information of common aspects of the task topic and that of the instance-specific aspects, both of which we are interested in discovering. We can make the sampling process of a word in question q dependent on the specific task instance it belongs to, making use of the task topic-instance structure available. Specifically, for the word w in task instance I_i, the sampling of w involves the background model of task topic (θ^T), k common aspect models

$(\theta_1^A, \cdots, \theta_k^A)$, and k instance-specific aspect models $(\theta_{1,i}^A, \cdots, \theta_{k,i}^A)$, used to capture the unique information about the k aspects in task instance I_i, i.e.

$$p_q(w|I_i) = \lambda_B p(w|\theta^T) + (1-\lambda_B)\sum_{j=1}^{k}\left[\pi_{q,j}\left(\lambda_C p(w|\theta_j^A) + (1-\lambda_C)p(w|\theta_{j,i}^A)\right)\right]$$

where $\pi_{q,j}$ is a question-specific mixing weight for the j-th aspect ($\sum_{j=1}^{k}\pi_{q,j} = 1$); λ_B is the mixing weight of the background model θ^T, which is set between 0 and 1; The purpose of using a background model is to "force" the clustering to be done based on more discriminative words, leading to more informative and more discriminative aspect theme models; λ_C is the weight on the common aspect model θ_j^A as opposed to the instance-specific aspect model $\theta_{j,i}^A$; λ_B and λ_C are set between 0 and 1; and the background model $p(w|\theta^T)$ is set as

$$p(w|\theta^T) = \frac{\sum_{i=1}^{m}\sum_{q\in I_i} c(w,q)}{\sum_{i=1}^{m}\sum_{q\in I_i}\sum_{w'\in V} c(w',q)} \tag{36}$$

which is used to "force" the clustering to be done based on more discriminative words, leading to more informative and more discriminative aspect theme models.

Intuitively, when we generate a word, we first decide whether to use the background model (θ^T) according to λ_B; a larger value of λ_B implies that the more likely we will use θ^T. If we decide not to use θ^T, then we need to decide which aspect theme to use; this is controlled by $\pi_{q,j}$, the probability of using j-th aspect theme when generating words in question q. Finally, once we decide which aspect theme to use, we further decide whether we use the common aspect model or the instance-specific aspect model, which is controlled by parameter λ_C. The weighting parameters λ_B and λ_C are intentionally to be set by the user, and the interpretation is as follows. λ_B reflects the belief about how noisy the questions under T are. If we believe the question text is usually verbose, then λ_B should be set to a larger value. In our experiments, a value of $0.9 - 0.95$ often works well. And, λ_C indicates our emphasis on

the commonality, as opposed to the specialty in comparative text mining. A larger value of λ_C would allow us to learn a richer common aspect model, whereas a smaller one would learn a weaker common aspect model, but stronger instance-specific aspect theme models. The optimal value would depend on the specific task topic we are interested in.

According to the model discussed above, the log-likelihood of the whole set of questions under the task topic $\mathcal{Q}^T$ is:

$$\log p(\mathcal{Q}^T|\Lambda) = \sum_{i=1}^{m} \sum_{q \in I_i} \sum_{w \in V} \{c(w,q) \log [\lambda_B p(w|\theta^T) + (1-\lambda_B) \sum_{j=1}^{k} \pi_{q,j} (\lambda_C p(w|\theta_j^A) + (1-\lambda_C) p(w|\theta_{j,i}^A))]\}$$

where λ_B and λ_C are set empirically; and the set of parameters to estimate, Λ, includes: (1) the common aspect models, $\{\theta_1^A, \cdots, \theta_k^A\}$; (2) the instance-specific aspect models, $\{\theta_{1,i}^A, \cdots, \theta_{k,i}^A\}$, for each task instance I_i under the task topic T, i.e. $(I_i \in \mathcal{I}(T))$; and (3) the mixing weights of aspect themes for each question q: $\{\pi_{q,1}, \cdots, \pi_{q,k}\}$. We can also apply EM algorithm to compute a maximum likelihood estimate. The updating formulas are shown in Figure 40. Each EM iteration involves scanning all the questions of the certain task topic once, so the algorithm is quite scalable.

8.2.2.2 Incorporating Aspect Priors

Recently, some CQA portals have explicitly provided expert knowledge about the task aspects. In particular, *eHow*, which is an online CQA portal dedicated to providing web users with the ability to research or share instructional solutions of task-related questions, has provided expert-defined aspects, for some of task topics and their task instances. For instance, "*hotel*", "*shopping*", and "*tourist attractions*" are three examples of aspects defined for the task topic "*travel*". In addition, eHow has also categorized some of user-generated questions into those specified aspects. In the

$$p(z_{q,I_i,w,j}) = \frac{(1-\lambda_B)\pi_{q,j}^{(n)}\left(\lambda_C p^{(n)}(w|\theta_j^A)+(1-\lambda_C)p^{(n)}(w|\theta_{j,i}^A)\right)}{\lambda_B p(w|\theta^T)+(1-\lambda_B)\sum_{j'=1}^{k}\pi_{q,j'}^{(n)}\left(\lambda_C p^{(n)}(w|\theta_{j'}^A)+(1-\lambda_C)p^{(n)}(w|\theta_{j',i}^A)\right)} \quad (37)$$

$$p(z_{q,I_i,w,B}) = \frac{\lambda_B p(w|\theta^T)}{\lambda_B p(w|\theta^T)+(1-\lambda_B)\sum_{j'=1}^{k}\pi_{q,j'}^{(n)}\left(\lambda_C p^{(n)}(w|\theta_{j'}^A)+(1-\lambda_C)p^{(n)}(w|\theta_{j',i}^A)\right)} \quad (38)$$

$$p(z_{q,I_i,w,j,C}) = \frac{\lambda_C p^{(n)}(w|\theta_j^A)}{\lambda_C p^{(n)}(w|\theta_j^A)+(1-\lambda_C)p^{(n)}(w|\theta_{j,i}^A)} \quad (39)$$

$$\pi_{q,j}^{(n+1)} = \frac{\sum_{w\in V} c(w,q)p(z_{q,I_i,w,j})}{\sum_{j'}\sum_{w\in V} c(w,q)p(z_{q,I_i,w,j'})} \quad (40)$$

$$p^{(n+1)}(w|\theta_j^A) = \frac{\sum_{i=1}^{m}\sum_{q\in I_i} c(w,q)p(z_{q,I_i,w,j})p(z_{q,I_i,w,j,C})}{\sum_{w'\in V}\sum_{i=1}^{m}\sum_{q\in I_i} c(w',q)p(z_{q,I_i,w',j})p(z_{q,I_i,w',j,C})} \quad (41)$$

$$p^{(n+1)}(w|\theta_{j,i}^A) = \frac{\sum_{q\in I_i} c(w,q)p(z_{q,I_i,w,j})(1-p(z_{q,I_i,w,j,C}))}{\sum_{w'\in V}\sum_{q\in I_j} c(w',q)p(z_{q,I_i,w',j})(1-p(z_{q,I_i,w',j,C}))} \quad (42)$$

Figure 40: EM updating formulas for the CTM-PLSA model

framework of probabilistic models, we can incorporate such expert knowledge for guiding the discovering of aspects by adding conjugate priors.

Specifically, we build a unigram language model $\{p(w|a_j)\}_{w\in V}$ for each pre-defined common aspect a_j based on the questions $\{q_1,\cdots,q_{n_j}\}$ which have been categorized into aspect a_j according to expert's prior knowledge. That is

$$p(w|a_j) = \frac{\sum_{s=1}^{n_j} c(w,q_s)}{\sum_{w'\in V}\sum_{s=1}^{n_j} c(w',q_s)}$$

Then, we could define a conjugate prior (i.e. Dirichlet prior) on each unigram language model, parameterized as

$Dir(\{\sigma_j p(w|a_j)+1\}_{w\in V})$, where σ_j is a confidence parameter for the prior. Since we use a conjugate prior, σ_j can be interpreted as the "equivalent sample size" because the effect of adding the prior would be equivalent to adding $\sigma_j p(w|a_j)$ pseudo counts for word w when we estimate the aspect theme model $p(w|\theta_j)$. Basically, the prior serves as some "training data" to intentionally bias the aspect discovering results. Similarly, we can also obtain the prior for each instance-specific aspect based on expert prior knowledge, i.e. $Dir(\{\sigma_{j,i}p(w|a_{j,i})+1\}_{w\in V})$.

To this end, the prior for all the parameters is given by

$$p(\Lambda) \propto \prod_{j=1}^{k+s_1} \prod_{w \in V} \left(p(w|\theta_j^A)^{\sigma_j p(w|a_j)} \times \prod_{i=1}^{m+s_2} p(w|\theta_{j,i}^A)^{\sigma_{j,i} p(w|a_{j,i})} \right)$$

where $\sigma_j = 0$ or $\sigma_{j,i} = 0$ if we do not have prior knowledge on some common aspect θ_j^A or instance-specific aspect $\theta_{j,i}^A$; and generally we have $s_1, s_2 > 0$, since we may want to find extra common aspects or instance-specific aspects other than the pre-defined aspects by expert knowledge.

Then, we can use the Maximum A Posteriori (MAP) estimator to estimate all the parameters as follows:

$$\hat{\Lambda} = \arg\max_{\Lambda} p(T|\Lambda)p(\Lambda)$$

The MAP estimate can be computed using essentially the same EM algorithm as presented above with slightly different updating formula for the component language models:

$$p^{(n+1)}(w|\theta_j^A) = \tag{43}$$

$$\frac{\sigma_j p(w|a_j) + \sum_{i=1}^{m} \sum_{q \in I_i} c(w,q) p(z_{q,I_i,w,j}) p(z_{q,I_i,w,j,C})}{\sigma_j + \sum_{w' \in V} \sum_{i=1}^{m} \sum_{q \in I_i} c(w',q) p(z_{q,I_i,w',j}) p(z_{q,I_i,w',j,C})} \tag{44}$$

$$p^{(n+1)}(w|\theta_{j,i}^A) = \tag{45}$$

$$\frac{\sigma_{j,i} p(w|a_{j,i}) + \sum_{q \in I_i} c(w,q) p(z_{q,I_i,w,j})(1 - p(z_{q,I_i,w,j,C}))}{\sigma_{j,i} + \sum_{w' \in V} \sum_{q \in I_j} c(w',q) p(z_{q,I_i,w',j})(1 - p(z_{q,I_i,w',j,C}))} \tag{46}$$

8.2.3 Generating Representative Description for Aspects

After the identification and clustering of aspects, we are trying to pull out some representative description for each common aspect and each instance-specific aspect in order to generate an integrated task summary for each task topic.

A straightforward method is to use a set of words with the highest probabilities within the common or instance-specific aspects. However, for better understanding of each instance-specific aspect, we seek to select a set of questions to form up a representative description. We observe that different questions under the same aspect may have different emphasis. For example, for the aspect "*hotel*" in task topic

"*travel*", questions of different users may emphasize different kinds of hotels, such as cheap hotels or hotels near the airport. In order to reveal the diversity of users' questions in the integrated task summary, we further cluster the questions under one instance-specific aspect into several groups. In detail, we use basic PLSA to do the clustering and set the number of groups proportional to the size of the aspect cluster. After that, one representative question in each group is selected that is closest to the cluster centroid (i.e. the word distribution) of the group; we use KL-Divergence of word distributions as the measure of distance.

If we define l groups $\{G_1, \cdots, G_l\}$ for the instance-specific aspect A, and extract the representative question q_i for group G_i, we have a representative description of aspect A as $\{q_1, q_2, \cdots, q_l\}$. We also present the best answers of the representative questions as the representative instructions for the group. As a result, we have the representative instructions of aspect A as $\{a^{q_1}, \cdots, a^{q_l}\}$, where a^{q_i} is the best answer of q_i.

8.3 Applications

The derived structured semantics of task questions can be utilized in several useful applications.

8.3.1 Identifying Semantics for New Questions

The extracted word distributions (i.e. the language model) of both common aspects and instance-specific aspects can be utilized to identify the semantics, i.e. the task topic, instance and aspect, of any new question. Specifically, given a new question q, we can recognize its semantics based on the similarity between the question language model θ_q and each mixing model which combines one instance-specific aspect model $\theta_{j,i}$ and the corresponding common aspect model θ_j. In this paper, we focus on the linear interpolation [32] (a.k.a linear smoothing) for combining two language models. Define an operator $\oplus_\lambda$ for linear smoothing where $a \oplus_\lambda b \equiv \lambda a + (1-\lambda)b$, assuming

a, b are both normalized to the same scale. When applied to two models, θ_1 and θ_2, we define that $\theta_1 \oplus_\lambda \theta_2$ as

$$p(w|\theta_1 \oplus_\lambda \theta_2) = \lambda p(w|\theta_1) + (1-\lambda)p(w|\theta_2), \quad \forall w \in V \tag{47}$$

We use KL Divergence as the similarity measure, thus we can identify q's semantics, i.e. assigning q to a specific task instance I_i^t and the aspect A_j^t of a given task topic T according to

$$\begin{aligned} &\arg\max_{j,i} - KL(\theta_q \| \theta_j^A \oplus_\lambda \theta_{j,i}^A) = \\ &\arg\max_{j,i} - \sum_{w \in V} p(w|\theta_q) \log \frac{p(w|\theta_q)}{p(w|\theta_j^A \oplus_\lambda \theta_{j,i}^A)} \end{aligned} \tag{48}$$

8.3.2 Finding Similar Questions

One of the most useful benefits of CQA services is that it provides the portal to automatically search previously asked questions, which can avoid the lag time involved with waiting for a personal response. Thus, another important application is about taking advantage of extracted structured semantics of archived questions to improve the performance retrieving similar questions for the new question. However, measuring semantic similarity between user-generated questions is not trivial. Since there are only a few words in each user-generated question and two questions that have the same meaning may use very different wording, traditional similarity measures developed for information retrieval may work poorly. In this section, we propose a method to incorporate the extracted aspects in structured semantics of task questions into the language modeling-based information retrieval.

8.3.2.1 Language Modeling Information Retrieval

In the language modeling (LM) approach to information retrieval (IR), queries (regarding new questions in the context of our paper) and documents (regarding archived questions in our paper) are modeled respectively by probabilistic LM. Let θ_q denote the parameters of a query (i.e. a new question) model and θ_d denote the parameters of

a document (i.e. an archived question) model. We apply a risk minimization framework for LM IR proposed earlier [42]. In this framework, the ranking is based on the measurement of the distance between two LMs: $\hat{\theta}_q$ and $\hat{\theta}_d$, which are obtained using maximum likelihood estimation observing the words in new question and archived questions. Here we employ KL Divergence to measure the distance $\Delta(\hat{\theta}_q, \hat{\theta}_d)$.

8.3.2.2 LM Expansion Using Aspect Theme Models

To improve the LM IR, we propose to incorporate the aspect analysis language models to improve retrieval quality, by expanding original query and document models, i.e. $\hat{\theta}_q \rightarrow \theta'_q$ as *query expansion*, and $\hat{\theta}_d \rightarrow \theta'_d$ as *document expansion*. We will apply the linear interpolation for combining two LMs as well. By using this combination method, one LM can be easily improved by smoothing with another "better" LM. In the following, we introduce two types of additional LMs we can estimate based on aspect analysis of archived questions. They expand the query and the document both with the extracted aspects.

1. Aspect-level query (new question) LMs

Since $\hat{\theta}_q$ is just the empirical distribution of the query q, the original word-level query model has shown to under-perform [42, 41], we seek to estimate the LMs at higher level. In particular, we consider each common or instance-specific aspect of any task topic as a token in the LM. All of these tokens comprise of a new vocabulary $V_a = \{v_1^a, \cdots, v_r^a\}$ and will later match the aspects discovered for the archived questions to determine their relevance.

First, we estimate the conditional probability that a word in new question belongs to each aspect v^a, say $p(v^a|w)$. Overall all aspects, we have a vector $\mathbf{v}_{V_a|w} = \langle p(v_1^a|w), \cdots, p(v_r^a|w)\rangle$, where r is the total number of aspect, either common or instance-specific over all task topics. After normalization, $\mathbf{v}_{V_a|w}$ becomes the probability distribution over all aspects, or rather, an aspect-level LM.

Then, we merge the multiple aspect distributions for each word in the new question into a single aspect distribution for the new question, denoted as γ_q^a. In the unigram case, γ_q^a is a vector of the same dimension with $\mathbf{v}_{V_a|w}$, and each element in γ_q^a denotes the probability of a particular aspect. Formally, we have $\gamma_q^a = \sum_{w \in q} \delta_w \mathbf{v}_{V_a|w}$, where δ_w is the normalized weight for the word w, and each element in γ_q^a denotes the probability of the corresponding aspect under this model. To represent a new question q, we use $\oplus_\lambda$ to combine the original word-level model θ_q with the aspect-level model γ_q^a.

2. Aspect-level document (archived question) LMs

Similar to the above approach, we could obtain the aspect distribution for each archived question, denoted as a vector $\gamma_d^a = \mathbf{v}_{V_a|d} = \langle p(v_1^a|d), \cdots, p(v_r^a|d) \rangle$. Considering this vector as a LM where each aspect is a unit, to represent an archived question d, we can use $\oplus_\lambda$ to combine the original word-level model θ_d with the aspect-level model γ_d^a.

8.3.3 Other Applications

Beyond the above two applications, there are still many additional ones that are useful for CQA services. One example is task-oriented diversified question recommendation. Unlike finding similar questions, the goal of this application is to diversify recommended results by suggesting task questions that belong to the same task instance but different task aspects with the user's new question. By recommending questions from other aspects of the same general task, this application could help the user to gain comprehensive knowledge of the general task he is taking.

Another useful application is to derive task-oriented user expertise. The goal is to specify the user expertise into the levels of task topic, task instance and task aspect. After that, if the asker submits a new task question, the CQA portal is able to suggest the asker which users have good reputation on the same task topic, instance or aspect,

Table 19: Examples of Integrated Task Summaries of two instances of The Task Topic "travel"

(a) Task Instance "travel to New York City"

Aspect	Common theme words	Instance-specific theme words	Representative questions	Representative instructions
Hotel	save, hotel, near, booking, rates, rooms, money, cheapest	city, hang, luxury, bar, famous, nyc, broadway, airport	How to find cheap hotels in New York City?	There is no such thing as a cheap room in NYC… Relatively affordable hotels in NYC are the Roosevelt, helmsley hotels…
			How to find a hotel near JFK airport in nyc?	Here are a couple from the airport's official web pages: http://www.panynj.gov/CommutingTravel/airports...
			How to find the best hotel bars in nyc?	I love the Atrium lounge in the Marriott Marquis in Times Square, vary spacious and…
Tourist attractions	park, tour, visit, tickets, pictures, water, advice, attractions	liberty, building, statue, natural, museum, area, trade, wall	How to visit the statue of liberty?	Take the 1 train south ferry, I think to the statue of liberty and Ellis islands the boat is 11.50 for adult…
			How to select the best museum in nyc?	The American museum of natural history also has a planetarium and extends along central park west for…It has the best dinosaur collection…
			How to see new york city in one day?	Empire state building, Coney island/astroland, ground zero, American museum of natural history…
Shopping	shopping, price, shop, souvenirs, currency, store, bargain, clothing	fashion, york, budget, soho, ave, city, cheap, nyc	How to find NYC fashion shopping guides?	Both this link: http://www.examiner.com/... And this link http://www.examiner.com/... will give you the location and links to stores…
			How to do shopping in NYC on a trip with a	my advice, hit clearance first, everything on sale. Also, look online before going and see which stores are having mega sales and plan purchases there…
			How to find cheap places for shopping in nyc?	I would recommend: Century 21, Find Outlet, Gabay's Outlet, Syms, Loehmann's…

(b) Task Instance "travel to Orlando"

Aspect	Common theme words	Instance-specific theme words	Representative questions	Representative instructions
Hotel	save, hotel, near, booking, rates, rooms, money, cheapest	rock, kids, fun, disney, park, mco, shuttle, family	How to find a hotel near all the theme park?	Its best to stay at a non resort on International Drive…
			How to find the best hotel in Orlando for kids?	You can try one of these: http://hotels.about.com/od/orlando/tp/kids_orlando.html
			How to find the closet hotel to Orlando airport (mco)?	There are a few hotels close to the airport there. There is actually a Hyatt located in the airport. …
Tourist attractions	park, tour, visit, tickets, pictures, water, advice, attractions	disney, theme, studios, price,sea, park, island, orlando	How to get tickets for disney Orlando in affordable price?	You should buy everything from the flight to Orlando,…and the disney word ticket at one price…try calling many agencies…
			How to pick a cheapest time to visit Disney in Orlando?	Disney world has four seasons, from least expensive to most expensive is value, regular, peek, and holiday…
			How to choose between universal studios or sea world or disney theme park?	We like all of them really. Universal has the best food…WDW is wonderful too. Not as much wow factor as USO…
Shopping	shopping, price, shop, souvenirs, currency, store, bargain, clothing	discount, outlet, teen, orlando, trip, best, world, downtown	How to find good shopping malls close to disney in orlando?	I like the Orlando premium outlets. It's off of vineland rd…there is also shopping off of international drive…
			How to get the best deal when doing some outlet shopping in orlando?	As josh above says either Prime Outlets or Premium Outlets. I also like Buena Vista Outlets…
			How to find the best teen shop stores in orlando?	There is Abercrombie, Abercrombie kids AE, Holloster… Most are either in the Millenia mall or the FL mall.

and to assign the new question to those experts, which could improve both efficiency and effectiveness of CQA services.

8.4 Experiments

In this section, we first introduce the dataset used in the experiment. Then, we demonstrate the effectiveness of our generative probabilistic modeling approach by using both empirical and quantitative evaluation methods. Furthermore, we provide evaluation on the effectiveness of the aspect analysis on external applications.

Table 20: Examples of Integrated Task Summaries of two instances of The Task Topic "pets"

(a) Task Instance "dogs as pets"

Aspect	Common theme words	Instance-specific theme words	Representative questions	Representative instructions
Health	vet, symptoms, fleas, disease, infection, care, clean, treat	dog, skin, puppy, allergies, diet, ear, pain, mites	How to treat dry skin of my dog?	After bathing your dog, work Aveeno bath oil into her skin…feed her salmon oil…
			How to make my dog eat a balanced diet?	He would need to be fed about 364 grams per day… make sure he has plenty of exercise…
			How to kill fleas on my dog?	…you can bathe with Dawn or Ivory dish soap…mix 1/2 cup of any vinegar or lemon juice in 1 gallon of warm water…
Training	training, command, behavior, use, treat, teach, potty, train	jump, stop, puppy, protect, hunting, house, collar, barking	How to train my dog to not jump?	…turning your back to them and ignoring them…tends to make them sit down…
			How to train a dog to become a hunting dog?	Buy a toy that looks like a squirrel or a duck that squeaks, then hide it in different places, praise him…
			How to train my dog to stop barking so much?	Get a tin can and put some rocks or marbles in it and when he starts barking shake the can and sternly say no…
Feeding	ingredient, best, feeding, make, oz, diet, food, raw	puppy, store, dog, homemade, brand, biscuits, again, meat	How to compare dog food ingredients?	Here you go: http://www.dogfoodanalysis.com/
			How to feed a nursing mother dog?	Lots of protein as in meat, cottage cheese and cooked eggs and make sure she has all the fresh clean water…
			How to buy the best dog food?	…high quality foods can be found at pet store chains, or online…a couple of foods I like are Nutro Natural, Innova, and Cannidae…

(b) Task Instance "birds as pets"

Aspect	Common theme words	Instance-specific theme words	Representative questions	Representative instructions
Health	vet, symptoms, fleas, disease, infection, care, clean, treat	nails, birds, injured, keep, active, bath, caged, nesting	How to care for an injured baby bird?	…you can use the following diet until a rehabilitator is found: 1 cup soaked canned food, 1/4 cup of applesauce, 1 hard boiled egg…
			How to clip my bird's nails?	Don't use the cement perches. an easier way to control the growth of the nails is by providing a large variety of perch shapes and sizes.
			How to take a bath to my pet bird?	...many different ways for birds to take baths…some cockatiels prefer misted with a spray bottle, others like bathing in a shallow dish…
Training	training, command, behavior, use, treat, teach, potty, train	parrot, talk, biting, speak, stop, your, hand, words	How to train my parrot to talk?	Repeat and repeat and repeat some more. 2-3 words and clearly…
			How to train my parrot to step up without biting?	…whenever he tries to bite you, don't go, just say very firmly "no bite!" and poit the finger at him…
			How to train a parrot/bird to land on your hand?	Food is your number one choice. I have birds of my own and its best to train them when they are young…
Feeding	ingredient, best, feeding, make, oz, diet, food, raw	birds, baby, feeder, seed, small, wild, garden, back	How to hand feed a baby bird?	...you would need to make up a gruel and syringe feed it…
			How to attract birds to my bird feeder?	Just keep it full of seed and give it time. It took about a month for the birds to really get used to our feeder…
			How to select the seed you feed to the birds in your yard?	I buy for specific birds I want to attract. Sunflower for Cardinals, thistle or Nyjer for Goldfinches, nutty suet for Woodpeckers…

8.4.1 Data Collection

In our experiment, we need two types of data sets for evaluation. One type is a set of user-generated questions related to certain task topic and task instance from CQA portals. We collect those questions from Yahoo! Answers.

Yahoo! Answers provides a hierarchical classification of all the question in its portal. We consider some higher level question categories as general task topics, such as "travel", "pets", "health" etc., while some lower level categories can be viewed as task instances of the certain task topic. For example, there are "travel to NYC",

Table 21: Basic statistics of the collections of task questions from Yahoo! Answers

Task Topic	# of instances	# of questions	# of words
Travel	6	2426	5483
Pets	6	2391	6664
Health	8	2045	7194

Table 22: Basic statistics of the prior knowledge of aspect from eHow

Task Topic	# of questions	# of words	# of aspects
Travel	300	956	15
Pets	200	834	10
Health	200	1118	10

"travel to Orlando", and "travel to Hawaii" under the task topic "travel"; for another example, there are "dogs", "cats", and "birds" under the task topic "pets". Based on this hierarchical classification, we can obtain the task topic-instance structure.

We utilize Yahoo! Answers API to collect task questions of the certain task topic. Specifically, we submit the phrase "how to" as query to the Yahoo! Answers Web services, with constraining the question category as one specific instance of the certain task topic, and retrieve up to 500 top-ranked related questions according to the Yahoo! Answers ranking. After retrieving questions for a set of instances of the task topic, we obtain a collection of task questions of the certain task topic. In order to ensure that retrieved questions are task questions, we also filter out those questions not including the exact phrase "how to". We conduct such collecting on three general task topics: "travel", "pets" and "health". Since there are so many instances under each task topic, we only collect a part of instances for each task topic. The basic information of these collections is shown in Table 21.

The other type of data is the prior knowledge of task aspects defined by experts. We construct this data set by leveraging the existing services provided by eHow. eHow also provides the hierarchical question classification, which is consistent with that of Yahoo! Answers. Thus, compared to Yahoo! Answers, we can find the same task topics and instances in eHow. Moreover, eHow supplies a set of task aspects

for each task topic, and each instance of this topic have the same aspects. Most of questions under one task instance in eHow are categorized into different task aspects of the corresponding task topic; while the other questions under this instance will not be assigned into any aspect.

To build the prior knowledge of each instance-specific aspect, we randomly crawl 20 questions assigned to each specific aspect under the certain task instance. Then, we collect all questions from each task instance of the certain task aspect together to form up the prior knowledge of the common aspect of the task topic. The composition and basic statistics of this dataset is shown in Table 22.

8.4.2 Empirical Results and Analysis

In this section, we evaluate the effectiveness of the CTM-PLSA on two general task topics: "travel" and "pets", empirically.

8.4.2.1 Task Topic I: Travel

In our data collection, the general task topic "travel" consists of six comparable task instances: "travel to New York City", "travel to Orlando", "travel to Hawaii", "travel to Las Vegas", "travel to San Francisco", and "travel to Chicago". Each of these task instances contains about 350 to 520 task questions. There are 15 pre-defined aspects based on the prior knowledge from experts in eHow. Our goal is to extract both common and instance-specific aspects of the task topic "travel" , and then generate the integrated task summary for each task instance. When computing the models, we empirically set $\lambda_B = 0.95, \lambda_C = 0.3$ for CTM-PLSA, and the number of clusters (i.e. number of aspects) to be 15. Variations of these parameters are discussed later.

Table 19 demonstrates part of integrated task summaries of two specific instances of the task topic "travel", which are obtained by using the CTM-PLSA model. Due to the limitation of the space, we illustrate only three aspects of these two instances. And

for each aspect, we show the top words of both common aspect models and instance-specific aspect models, along with the corresponding representative questions. To generate representative questions for each instance-specific aspect, we further cluster all questions assigned to the certain instance-specific aspect into 5 groups and show 3 of them with top 3 largest sizes in the table. Moreover, we also show the major part (sentences) of the representative instructions.

From the table, we find that the results of CTM-PLSA explicitly suggest the common aspect themes and the corresponding instance-specific aspect themes. In particular, the top words of the common aspect themes can effectively represent the semantics of the common aspects, without suggesting any specific task instance. For example, the words "hotel", "booking" and "rooms" provide a good description of the semantics of the common aspect "hotel", but none of them is related to the instance either "New York City" or "Orlando". Furthermore, the top words of the instance-specific aspect themes include key words better describing the semantics of the aspects in the context of the task instances. For example, the top words of "hotel" aspect in the "New York City" instance include "broadway" which is a famous street there and "nyc" which is the brevity of that city; while in comparison, the top words in the corresponding "Orlando" instance include "disney" which is the famous theme park in Orlando and "mco" which is the brevity of Orlando airport. And we can observe the similar cases on the other common and instance-specific aspects.

Digging into the representative questions and instructions of each aspect, we can also see that there is indeed some interesting information discovered. In particular, (1) for the "hotel" aspect, web users are often interested in hotels near the airport, in the context of either "New York City" or "Orlando"; users tend to find cheap hotels in New York City due to the high price there, whereas users aim to find those best for kids since many parents take their children to visit Disney Park in Orlando. (2) for the "tourist attractions" aspect, people usually hope to look around several famous

spots in New York City, such as the Statue of Liberty and some museums, while parents and their kids seek to have happy time in Disney and other theme parks in Orlando. (3) and for the "shopping" aspect, many users hope to find cheap stores in both context; but some of users look for shops for fashion in New York City, while few people search fashion in Orlando.

8.4.2.2 Task Topic II: Pets

The general task topic "pets", in our data collection, consists of six comparable task instances: "cats", "dogs", "birds", "horses", "fishes", and "reptile". Each of these task instances contains about 330 to 480 task questions. There are 10 pre-defined aspects based on the prior knowledge from experts in eHow. Similar to the last example, our goal is to extract both common and instance-specific aspects of the task topic "pets", and generate the integrated task summary for each task instance. When computing the models, we empirically set $\lambda_B = 0.95, \lambda_C = 0.45$ for CTM-PLSA, and the number of aspects to be 10.

Table 20 demonstrates a part of integrated task summaries of two instances ("dogs" and "birds"), which are obtained by using CTM-PLSA. Similar to the last example, we illustrate only three aspects of these two instances; and for each aspect, we show the top words of both common and instance-specific aspect models, along with the corresponding representative questions. We use the method same to last example to extract the representative questions, and we also show the major part (sentences) of the representative instructions.

The result on this task topic are generally similar to those on "travel" topic. From the table, we observe that the results of CTM-PLSA explicitly suggest the common aspect themes and the corresponding instance-specific aspect themes. In particular, the words "training", "command" and "teach" can indicate the semantics of the common aspect "training" well, but none of them is related to the instance either

"dogs" or "birds". However, the top words of "training" aspect in the "dogs" instance include "jump", "hunting", and (*not*) "barking" which are the apparent evidence for the semantics of training dogs; in comparison, the top words in the corresponding "birds" instance include "talk", "speak" and probably *not* "biting" which are also the obvious indication of the semantics of training birds. And we can observe the similar cases on the other common and instance-specific aspects.

We can similarly discover some interesting information from the representative questions and instructions of each aspect. In particular, in addition to "training" aspect as discussed above, (1) for the "health" aspect, fleas and dry skin are two serious problem to the health of dogs while birds have different concern on health, such as injury of baby birds. (2) for the "feeding" aspect, users are usually careful with the ingredients and quality of dog food, while for birds, people are more interested in how to attract birds to the feeder or how to hand feed them.

8.4.2.3 Parameter Tuning

In the CTM-PLSA model, we can generate different results when varying the values of parameters λ_B and λ_C. Specifically, if we set λ_B to a smaller value, non-informative stop words tend to show up in common aspect themes with higher probabilities. According the empirical experiments, a reasonable value for λ_B would be generally higher than 0.9, which can help to automatically eliminate the non-informative words from the aspect themes and to allow for more discriminative aspects. From Table 19 and 20, we can see that the model is clearly able to filter out most of non-informative words. Even higher value of λ_B may help to *eliminate* non-informative words, but it may cause the insufficient information for us to learn a common aspect reliably.

The value of parameter λ_C affects the vocabulary allocation between the common aspects and instance-specific aspects. In the experiments on "travel" topic, when we change λ_C to a value above 0.5, some instance-specific terms would show up in

common aspect themes; while in the experiments on "pets" topic, when λ_C is less than 0.3, many keywords of the common aspect themes are shown up in the corresponding instance-specific aspect themes.

8.4.3 Quantitative Evaluation

In order to quantitatively evaluate the effectiveness of aspect discovery and clustering, we ask users to manually group questions into aspect clusters as our gold standard. In order to reduce the bias, we collect the evaluation results from three users, who are all graduate students. For each task instance, we first select 100 most popular questions ranked by the number of associated answers. Then, for each of those 100 questions, the three users are asked to assign one of the aspects defined by eHow to it. In essence, this is a multi-class classification problem where the number of classes is the number of aspects of the specific task instance defined by eHow.

Table 23 reports the number of questions whose aspect labels are agreed on among the three users with respect to different task instances or task topics. It shows that more than 70% questions are assigned into the same aspect for each task instance or task topic. In the following test, we use all the questions with agreed aspect labels from three annotators as the gold standard.

Table 23: Inter-annotator agreement for aspect discovering

	travel:NYC	*travel:Orlando*	*travel:overall*
Agreement	72%	76%	73%
	pets:dogs	*pets:birds*	*cats:overall*
Agreement	83%	75%	77%

We propose to use *Clustering Accuracy* to measure the clustering coherence performance for aspect discovery. Given a question q, let $H(q)$ and $L(q)$ be the human annotated aspect label and the label generated by some algorithm, respectively. The clustering accuracy is defined as follows:

$$\text{ClusteringAccuracy} = \frac{\sum_{q \in Q} \delta(H(q), map(L(q)))}{|Q|}$$

where $|Q|$ is the total number of questions, $\delta(x, y)$ is the delta function that equals one is $x = y$ and equals zeros otherwise, and $map(L(q))$ is the permutation mapping function that maps each cluster label $L(q)$ to the equivalent label from the human annotation. The best mapping can be found by using the Kuhn-Munkres algorithms [48].

For each method for aspect discovery, we compute the Clustering Accuracy of each task instance and take the average over all task instances as the performance of the specific method. As shown on Table 24, CTM-PLSA with incorporated aspect priors achieves the best performance of Clustering Accuracy, meaning that the obtained aspects are most coherent with respect to human generated aspects. This is consistent with our analysis in Section 8.2.2.

Table 24: Evaluation of Clustering Accuracy

Method	Clustering Accuracy
k-means	0.35
Basic PLSA	0.43
CTM-PLSA	0.62
CTM-PLSA with aspect priors	0.74

8.4.4 Experiments on External Applications

8.4.4.1 Identifying Semantics for New Questions

In this experiment, we seek to evaluate whether the extracted aspect models based on CTM-PLSA can be effective to identify the semantics for new coming questions. Specifically, we randomly collect 100 questions for each task topic ("travel" and "pets") from eHow. And, eHow has provided the structured semantics in terms of task instance and aspect for each of these questions, which can be considered as the gold standard, i.e. true assignment of task instance and aspect, for these new questions. Note that these 100 questions are different from those used as the expert prior knowledge for extracting structured semantics. In the experiment, we adapt the metric of overall accuracy, measuring the fraction of questions which are assigned into

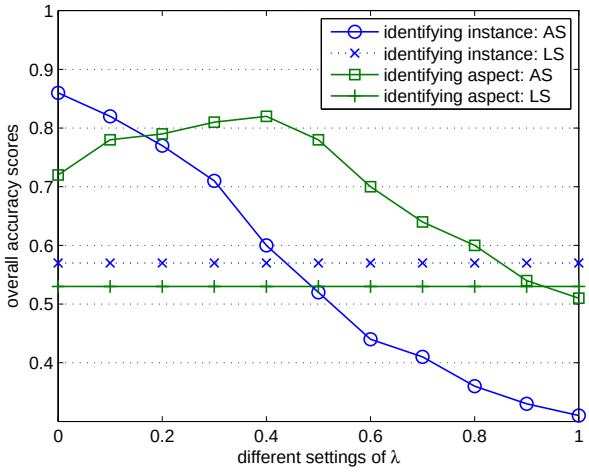


Figure 41: The overall accuracy of instance and aspect identification for new questions

the correct instance or aspect, to evaluate the performance of semantic identification for new questions.

To evaluate the performance of identifying the task instances (or aspects) for new questions, we compare the following methods:

•**LM similarity (LS):** For each task instance (aspect), we use all questions (from those used as priors in CTM-PLSA) which are pre-defined as in this task instance (aspect) to build the LM θ_I (θ_A) for this task instance (aspect). Given a new question with its LM θ_q, we identify its task instance (aspect) according to $\arg\max_I KL(\theta_I \| \theta_q)$ ($\arg\max_A KL(\theta_A \| \theta_q)$), where $KL(\cdot\|\cdot)$ measures the KL Divergence between two LMs.

•**Aspect analysis (AS):** Based on the aspect analysis of CTM-PLSA model, we use Eq. 48 to identify the task instance (aspect) for the new question.

Figure 41 demonstrates the overall accuracy of task instance and aspect identification over all of 300 new questions of three task topics. This figure also illustrates the change of accuracy scores for *aspect analysis* method against different settings of λ, which is the mixing weight between the common aspect model and instance-specific aspect model.

From the figure, we can see that the accuracy of instance and aspect identification

is sensitive to λ. For instance identification, the accuracy score decreases as the value of λ increases. Thus, instance-specific model is more crucial to instance identification. And AS method can achieve better performance than LS method when $\lambda < 0.4$. For aspect identification, λ reaches best performance at about 0.4. AS method can achieve better performance than LS method when $\lambda < 0.9$. Moreover, when $\lambda \to 1$, the common aspect model itself indicates the similar accuracy with LS method on identifying aspect for new questions. Therefore, mixing instance-aspect model with common aspect model will improve the accuracy of aspect identification.

8.4.4.2 Finding Similar Questions

Now we evaluate the performance of similar question retrieval of various LM approaches. The methods we compare are:

•**Word-level LM (WLM)**: LMs of archived and new questions are defined by the original text of respective questions.

•**Word-level + basic Aspect-level LM (WALM-basic)**: For each task topic, we run the basic PLSA over all instances, obtaining the aspects of all questions under the specific task topic. Then, the aspect-level LM is combined with the word-level LM (WLM), using parameter λ.

•**Word-level + cross Aspect-level LM (WALM-cross)**: For each task topic, we run the CTM-PLSA without any prior (Sec 8.2.2.1) over all instances, obtaining both common and instance-specific aspects. Then, we use common aspects to generate the aspect-level LM, and combine it with WLM, using parameter λ.

•**Word-level + prior Aspect-level LM (WALM-prior)**: During training WALM-cross, we incorporate expert prior knowledge on aspects into CTM-PLSA. Then, we combine aspect-level LM with WLM, using parameter λ. Moreover, we implement the word translation model proposed in a related work [33], which is defined as:

•**Word translation model (WTM)**: As proposed in [33], we retrieve similar questions based on a word translation model implemented using a collection of archived questions.

For evaluation, we randomly collect 90 new questions under the three task topics "travel","pets","health" from Yahoo! Answer, but not included in the training set. Each topic is associated with 30 new questions. Then, for each new question, we apply the above four methods for retrieving similar archived questions. The quality of retrieval is evaluated on the top 5 archived questions using *Precision at* K ($P(K)$), which reports the fraction of questions ranked in the top K results that are labeled as relevant. Two human judges are invited to provide feedback on the composite set of questions which occur in any of the top 5 retrieval results. Each judgment is of two levels, either *relevant* or *irrelevant*. And judgments are carried out independently based on their experience of the relevance quality. In general the judgments made by two persons results in over 90% agreement. For those they have inconsistent judgments, we randomly assign *relevant* or *irrelevant* labels. To determine the optimal λ for three WALM based methods, we conduct 5 fold cross-validation against user judgment.

Figure 42 illustrates the Precision at K of the compared methods. From the figure, we can see that both the translation model and three WALM based methods outperform the basic WLM method, demonstrating that the use of aspect information can improve the quality of question retrieval. We can also observe that WALM-cross and WALM-prior can achieve better performance than WALM-basic, which indicates that it is more beneficial to extract questions' semantics by applying CTM-PLSA to obtain both common and instance-specific aspects rather than using basic PLSA. Furthermore, the figure shows that WALM-prior implies higher quality of questions retrieval than WALM-cross, which indicates that it is more effective to extract questions' semantics by incorporating expert prior knowledge on aspects when

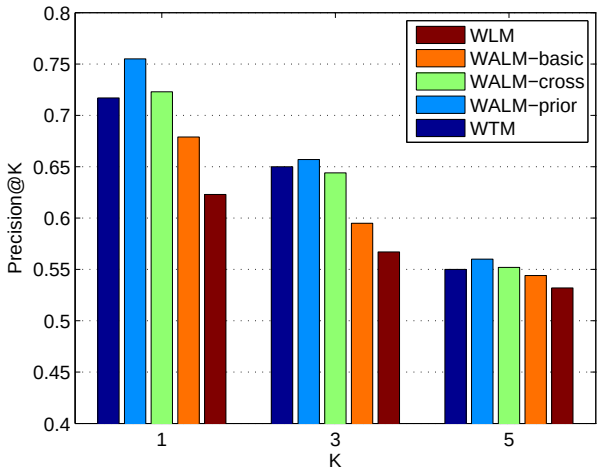


Figure 42: Precision at K for four compared methods: WLM, WALM-basic, WALM-cross, WALM-prior, WTM

training CTM-PLSA, such that the extracted semantics could be more beneficial to improve the quality of question retrieval. It is also worthwhile to mention that WALM-prior can improve the performance of a very recent related work [33] (WTM). After conducting t-test, we find that the improvements of WALM-prior over other methods are statistically significant (p-value$<$ 0.05).

8.5 Summary

In this chapter, we formally define the problem of extracting structured semantics of task questions and generating integrated task summaries based on such structured semantics. We address this problem by proposing to use a generative topic model for comparative text mining, and we also incorporate prior expert knowledge into the learning process. The results of empirical experiments show that the proposed method is effective for both extracting structured semantics of questions and generating integrated task summaries for each task topic. We also evaluate our method by studying the effectiveness of using extracted aspects of questions to improve external applications, including identifying semantics and retrieving similar questions for new questions. The experimental results demonstrate that the context information

of structured semantics can benefit the search performance significantly.

CHAPTER IX

CONCLUSION

This research consists of a series of new algorithms and techniques for leveraging rich context information of both the query and the Web content to improve search performance. In general, the context information of the query can be used to specify different ranking objectives for various queries, based on which new methods are proposed to formalize new global objective function so as to improve the ranking relevance; new techniques for extracting various context information of the Web content, such as content quality, author's reputation, and user's interactions, are crucial to build an effective search system over the specific form of Web content, such as social media service.

In particular, in terms of exploring the context information of the query, contributions of this dissertation include: (1) New method for incorporating query difference into constructing ranking models by introducing query-dependent loss functions [7]. Based on a popular query taxonomy of Web search, the new position-sensitive query-dependent loss function for ranking is exploited and applied on two concrete ranking algorithms. Moreover, beyond the straightforward method learning ranking model with pre-defined query categorization, a new learning method is introduced to conduct learning of the ranking model jointly with that of query categorization. (2) Inspired by the difficulty in defining query difference of more query categories in the loss function and the requirement of deep dive and incremental update on dedicated ranking models, we explore a new divide-and-conquer approach to learn multiple ranking functions according to diverse ranking characteristics of queries [6]. This approach consists of three major steps: identification of ranking-sensitive query topics,

a unified learning process to obtain ranking models corresponding to recognized query topics, and an ensemble method for computing relevance for new queries.

Future research on exploring the context of the query include: (1) Deeper exploration on what kind of features as well as which aggregation method are more essential to identify ranking-sensitive topics for queries. (2) Investigation on how to recognize the ranking-sensitive query topics jointly with learning the ranking function.

Furthermore, in terms of exploring the context information of the Web content, contributions of this dissertation include: (1) New robust and effective framework for retrieving social media content. The study has been complemented with an analysis of the results to gain insight into the significant dimensions of fact retrieval from social media. (2) Exploration of a new framework for semi-supervised quality and reputation estimation of content and users in social media. It is also demonstrated that using the obtained quality and reputation information can result in significant improvement for the practical task of searching the social media content. (3) Investigation on the *robustness* of ranking in the presence of malicious feedback (vote spam), analyzation on general models for common vote spam strategies, and development on a training method that improves the robustness of ranking by injecting simulated spam into the training data. (4) Extraction and exploration of a new type of context information describing the structured semantics of social media content. The studies on information retrieval applications demonstrate that this new type of context information can benefit the search performance significantly.

Future work on exploring the context of social media content includes: (1) Exploration on new types of context information, such as time dynamics, location information, etc., for new emerging forms of social media service, including microblogging, social recommendation service, etc. (2) Investigation on how to take advantage of information from other different social media services as the context information to enhance the search performance in one specific social media service.

REFERENCES

[1] Adamic, L. A., Zhang, J., Bakshy, E., and Ackerman, M. S., "Knowledge sharing and yahoo answers: everyone knows something," in *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pp. 665–674, ACM, 2008.

[2] Agichtein, E., Brill, E., and Dumais, S., "Improving web search ranking by incorporating user behavior information," in *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 19–26, ACM, 2006.

[3] Agichtein, E., Castillo, C., Donato, D., Gionis, A., and Mishne, G., "Finding high-quality content in social media," in *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pp. 183–194, ACM, 2008.

[4] Baeza-Yates, R. A. and Ribeiro-Neto, B., *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., 1999.

[5] Beitzel, S. M., Jensen, E. C., Chowdhury, A., and Frieder, O., "Varying approaches to topical web query classification," in *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 783–784, ACM, 2007.

[6] Bian, J., Li, X., Li, F., Zheng, Z., and Zha, H., "Ranking specialization for web search: a divide-and-conquer approach by using topical ranksvm," in *WWW '10: Proceedings of the 19th international conference on World wide web*, pp. 131–140, ACM, 2010.

[7] Bian, J., Liu, T.-Y., Qin, T., and Zha, H., "Ranking with query-dependent loss for web search," in *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*, pp. 141–150, ACM, 2010.

[8] Bian, J., Liu, Y., Agichtein, E., and Zha, H., "A few bad votes too many?: towards robust ranking in social media," in *AIRWeb '08: Proceedings of the 4th international workshop on Adversarial information retrieval on the web*, pp. 53–60, ACM, 2008.

[9] Bian, J., Liu, Y., Agichtein, E., and Zha, H., "Finding the right facts in the crowd: factoid question answering over social media," in *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pp. 467–476, ACM, 2008.

[10] Bian, J., Liu, Y., Zhou, D., Agichtein, E., and Zha, H., "Learning to recognize reliable users and content in social media with coupled mutual reinforcement," in *WWW '09: Proceedings of the 18th international conference on World wide web*, pp. 51–60, ACM, 2009.

[11] Bian, J., Lu, Y., and Zha, H., "Turning social media content into task knowledge: Structured semantics extraction for task questions," in *In submission*, 2010.

[12] Blei, D. M., Ng, A. Y., and Jordan, M. I., "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.

[13] Bolstad, B., Irizarry, R., Astrand, M., and Speed, T., "A comparison of normalization methods for high density oligonucleotide array data based on variance and bias," *Bioinformatics*, vol. 19, no. 2, pp. 185–193, 2003.

[14] Brill, E., Dumais, S., and Banko, M., "An analysis of the askmsr question-answering system," in *In Proceedings of 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 257–264, 2002.

[15] Broder, A., "A taxonomy of web search," *SIGIR Forum*, vol. 36, no. 2, pp. 3–10, 2002.

[16] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G., "Learning to rank using gradient descent," in *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pp. 89–96, ACM, 2005.

[17] Burke, R. D., Hammond, K. J., Kulyukin, V. A., Lytinen, S. L., Tomuro, N., and Schoenberg, S., "Question answering from frequently-asked question files: Experiences with the faq finder system," tech. rep., AI Magazine, 1997.

[18] Campbell, C. S., Maglio, P. P., Cozzi, A., and Dom, B., "Expertise identification using email communications," in *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pp. 528–531, ACM, 2003.

[19] Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H., "Learning to rank: from pairwise approach to listwise approach," in *ICML '07: Proceedings of the 24th international conference on Machine learning*, pp. 129–136, ACM, 2007.

[20] Caruana, R., "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.

[21] Dom, B., Eiron, I., Cozzi, A., and Zhang, Y., "Graph-based ranking algorithms for e-mail expertise analysis," in *DMKD '03: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pp. 42–48, ACM, 2003.

[22] Freund, Y., Iyer, R., Schapire, R. E., and Singer, Y., "An efficient boosting algorithm for combining preferences," *J. Mach. Learn. Res.*, vol. 4, pp. 933–969, 2003.

[23] Friedman, J. H., "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, pp. 1189–1232, 1999.

[24] Geng, X., Liu, T.-Y., Qin, T., Arnold, A., Li, H., and Shum, H.-Y., "Query dependent ranking using k-nearest neighbor," in *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 115–122, ACM, 2008.

[25] Guha, R., Kumar, R., Raghavan, P., and Tomkins, A., "Propagation of trust and distrust," in *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pp. 403–412, ACM, 2004.

[26] Herbrich, R., Graepel, T., and Obermayer, K., "Support vector learning for ordinal regression," in *In International Conference on Artificial Neural Networks*, pp. 97–102, 1999.

[27] Heymann, P., Koutrika, G., and Garcia-Molina, H., "Fighting spam on social web sites: A survey of approaches and future challenges," *IEEE Internet Computing*, vol. 11, no. 6, pp. 36–45, 2007.

[28] Hofmann, T., "Unsupervised learning by probabilistic latent semantic analysis," *Mach. Learn.*, vol. 42, no. 1-2, pp. 177–196, 2001.

[29] Immorlica, N., Jain, K., Mahdian, M., and Talwar, K., "Click fraud resistant methods for learning click-through rates," pp. 34–45, 2005.

[30] Jansen, B. J., "Adversarial information retrieval aspects of sponsored search.," in *AIRWeb*, pp. 33–36, 2006.

[31] Järvelin, K. and Kekäläinen, J., "Cumulated gain-based evaluation of ir techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, 2002.

[32] Jelinek, F. and Mercer, R. L., "Interpolated estimation of markov source parameters from sparse data," in *Proceedings of the Workshop on Pattern Recognition in Practice*, (Amsterdam, The Netherlands: North-Holland), May 1980.

[33] Jeon, J., Croft, W. B., and Lee, J. H., "Finding similar questions in large question and answer archives," in *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pp. 84–90, ACM, 2005.

[34] Jeon, J., Croft, W. B., Lee, J. H., and Park, S., "A framework to predict the quality of answers with non-textual features," in *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 228–235, ACM, 2006.

[35] Joachims, T., "Optimizing search engines using clickthrough data," in *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 133–142, ACM, 2002.

[36] Joachims, T., Granka, L., Pan, B., Hembrooke, H., and Gay, G., "Accurately interpreting clickthrough data as implicit feedback," in *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 154–161, ACM, 2005.

[37] Jurczyk, P. and Agichtein, E., "Discovering authorities in question answer communities by using link analysis," in *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pp. 919–922, ACM, 2007.

[38] Kang, I.-H. and Kim, G., "Query type classification for web document retrieval," in *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pp. 64–71, ACM, 2003.

[39] Kelly, D. and Teevan, J., "Implicit feedback for inferring user preference: a bibliography," *SIGIR Forum*, vol. 37, no. 2, pp. 18–28, 2003.

[40] Kleinberg, J. M., "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[41] Kurland, O., Lee, L., and Domshlak, C., "Better than the real thing?: iterative pseudo-query processing using cluster-based language models," in *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 19–26, ACM, 2005.

[42] Lafferty, J. and Zhai, C., "Document language models, query models, and risk minimization for information retrieval," in *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 111–119, ACM, 2001.

[43] Lee, U., Liu, Z., and Cho, J., "Automatic identification of user goals in web search," in *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pp. 391–400, ACM, 2005.

[44] Lenz, M., Hübner, A., and Kunze, M., "Question answering with textual cbr," in *FQAS '98: Proceedings of the Third International Conference on Flexible Query Answering Systems*, pp. 236–247, Springer-Verlag, 1998.

[45] Li, P., Burges, C. J. C., and Wu, Q., "Mcrank: Learning to rank using multiple classification and gradient boosting.," in *NIPS*, MIT Press, 2007.

[46] Li, W. and McCallum, A., "Pachinko allocation: Dag-structured mixture models of topic correlations," in *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pp. 577–584, ACM, 2006.

[47] Liu, T. Y., Xu, J., Qin, T., Xiong, W., and Li, H., "Letor: Benchmark dataset for research on learning to rank for information retrieval," in *SIGIR '07: Proceedings of the Learning to Rank workshop in the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007.

[48] Lovasz, L. and Plummer, M., "Matching theory," in *Annals of Discrete Mathematics*, 1986.

[49] Lu, Y. and Zhai, C., "Opinion integration through semi-supervised topic modeling," in *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pp. 121–130, ACM, 2008.

[50] Lu, Y., Zhai, C., and Sundaresan, N., "Rated aspect summarization of short comments," in *WWW '09: Proceedings of the 18th international conference on World wide web*, pp. 131–140, ACM, 2009.

[51] Marx, Z., Dagan, I., Buhmann, J. M., and Shamir, E., "Coupled clustering: a method for detecting structural correspondence," *J. Mach. Learn. Res.*, vol. 3, pp. 747–780, 2003.

[52] McCallum, A. K., "Multi-label text classification with a mixture model trained by em," 1999.

[53] Mehta, B., Hofmann, T., and Fankhauser, P., "Lies and propaganda: detecting spam users in collaborative filtering," in *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, pp. 14–21, ACM, 2007.

[54] Mei, Q., Ling, X., Wondra, M., Su, H., and Zhai, C., "Topic sentiment mixture: modeling facets and opinions in weblogs," in *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pp. 171–180, ACM, 2007.

[55] Mei, Q., Liu, C., Su, H., and Zhai, C., "A probabilistic approach to spatiotemporal theme pattern mining on weblogs," in *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pp. 533–542, ACM, 2006.

[56] Mei, Q. and Zhai, C., "A mixture model for contextual text mining," in *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 649–655, 2006.

[57] Metwally, A., Agrawal, D., and El Abbadi, A., "Detectives: detecting coalition hit inflation attacks in advertising networks streams," in *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pp. 241–250, ACM, 2007.

[58] National, E. V. and Voorhees, E. M., "Overview of the trec 2003 question answering track," pp. 54–68, 1999.

[59] Page, L., Brin, S., Motwani, R., and Winograd, T., "The pagerank citation ranking: Bringing order to the web.," Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[60] Ponte, J. M. and Croft, W. B., "A language modeling approach to information retrieval," in *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 275–281, ACM, 1998.

[61] Radlinski, F., "Addressing malicious noise in clickthrough data," in *AIRWeb*, 2007.

[62] Radlinski, F. and Joachims, T., "Minimally invasive randomization for collecting unbiased preferences from clickthrough logs," in *AAAI'06: proceedings of the 21st national conference on Artificial intelligence*, pp. 1406–1412, AAAI Press, 2006.

[63] Ramakrishnan, R. and Tomkins, A., "Toward a peopleweb," *Computer*, vol. 40, pp. 63–72, 2007.

[64] Robertson, S., "Overview of the okapi projects," *Journal of Documentation*, vol. 53, no. 1, pp. 3–7, 1998.

[65] Rose, D. E. and Levinson, D., "Understanding user goals in web search," in *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pp. 13–19, ACM, 2004.

[66] Rosen-Zvi, M., Griffiths, T., Steyvers, M., and Smyth, P., "The author-topic model for authors and documents," in *UAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pp. 487–494, AUAI Press, 2004.

[67] Salton, G. and Lesk, M. E., "Computer evaluation of indexing and text processing," *J. ACM*, vol. 15, no. 1, pp. 8–36, 1968.

[68] Sarawagi, S., Chakrabarti, S., and Godbole, S., "Cross-training: learning probabilistic mappings between topics," in *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 177–186, ACM, 2003.

[69] Scott, J., *Social Network Analysis: A Handbook*. Sage Publications, second. ed., 2000.

[70] Shen, D., Sun, J.-T., Yang, Q., and Chen, Z., "Building bridges for web query classification," in *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 131–138, ACM, 2006.

[71] Sneiders, E., "Automated faq answering: Continued experience with shallow language understanding," 1999.

[72] Soricut, R. and Brill, E., "Automatic question answering using the web: Beyond the factoid," *Inf. Retr.*, vol. 9, no. 2, pp. 191–206, 2006.

[73] Steyvers, M., Smyth, P., Rosen-Zvi, M., and Griffiths, T., "Probabilistic author-topic models for information discovery," in *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 306–315, ACM, 2004.

[74] Su, Q., Pavlov, D., Chow, J.-H., and Baker, W. C., "Internet-scale collection of human-reviewed data," in *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pp. 231–240, ACM, 2007.

[75] Tibshirani, R., Walther, G., and Hastie, T., "Estimating the number of clusters in a dataset via the gap statistic," vol. 63, pp. 411–423, 2000.

[76] Voorhees, E. M. and Harman, D. K., *TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic Publishing)*. The MIT Press, 2005.

[77] Xia, F., Liu, T.-Y., Wang, J., Zhang, W., and Li, H., "Listwise approach to learning to rank: theory and algorithm," in *ICML '08: Proceedings of the 25th international conference on Machine learning*, pp. 1192–1199, ACM, 2008.

[78] Zha, H., "Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering," in *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 113–120, ACM, 2002.

[79] Zha, H., Zheng, Z., Fu, H., and Sun, G., "Incorporating query difference for learning retrieval functions in world wide web search," in *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pp. 307–316, ACM, 2006.

[80] Zhai, C., Velivelli, A., and Yu, B., "A cross-collection mixture model for comparative text mining," in *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 743–748, ACM, 2004.

[81] Zhang, J., Ackerman, M. S., and Adamic, L., "Expertise networks in online communities: structure and algorithms," in *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pp. 221–230, ACM, 2007.

[82] Zheng, Z., Chen, K., Sun, G., and Zha, H., "A regression framework for learning ranking functions using relative relevance judgments," in *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 287–294, ACM, 2007.

[83] Zheng, Z., Zha, H., and Sun, G., "Query-level learning to rank using isotonic regression," in *Allerton '08: Proceedings of the 46th Annual Allerton Conference on Communication, Control and Computing*, 2008.

[84] Zhou, D., Bian, J., Zheng, S., Zha, H., and Giles, C. L., "Exploring social annotations for information retrieval," in *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pp. 715–724, ACM, 2008.

[85] Zhou, D., Manavoglu, E., Li, J., Giles, C. L., and Zha, H., "Probabilistic models for discovering e-communities," in *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pp. 173–182, ACM, 2006.

[86] Zhou, D., Orshanskiy, S. A., Zha, H., and Giles, C. L., "Co-ranking authors and documents in a heterogeneous network," in *ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, pp. 739–744, IEEE Computer Society, 2007.

[87] Ziegler, C.-N. and Lausen, G., "Propagation models for trust and distrust in social networks," *Information Systems Frontiers*, vol. 7, no. 4-5, pp. 337–358, 2005.

VITA

Jiang Bian was born in Beijing, the capital of China. He received his bachelor's degree in Computer Science from Peking University in Beijing in July 2006. In August 2006, he enrolled in the Ph.D. program in Computer Science at Georgia Institute of Technology where he developed research in exploring contextualized Web search and engaged in a wide range of projects involving information retrieval, social network analysis, learning to rank, and text mining over various Web environments, such as generic Web search, search over the community questions answering, social bookmarking, and microblogging. During his Ph.D. study, he has published several papers in many venues including conferences for information retrieval and data mining.

His research has received attention and collaboration with both industry and academia. He interned at Nokia, Microsoft, Yahoo!, and Facebook in 2007, 2008, 2009, and 2010, respectively, and collaborated with professors and students from Emory University, University of Illinois at Urbana-Champaign and University of California at Irvine. He will join Yahoo! Labs as a research scientist from Oct. 2010.